ON ADAPTIVE CONTROL OF ROBOTS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

PRASHANT VIJAY WAKNIS

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

FEBRUARY, 1988

CERTIFICATE

Certified that the work embodied in this thesis entitled 'On Adaptive Control of Robots' has been carried out by Mr. Prashant Vijay Waknis under my supervision and the same has not been submitted elsewhere for a degree.

(Arindam Ghosh)
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology
KANPUR.

ACKNOWLEDGEMENTS

My heartfelt gratitude to my thesis supervisor, Dr. Arindam Ghosh for his unflagging patience, understanding and encouragement and his generous sharing of ideas which made this work a true amalgam of business and pleasure.

Thanks to Sanjeev Khadilkar for the fruitful discussions throughout the project work, to B.R. Patnaik for his open hearted cooperation. Thanks too, to the incomparable batch of '87 for all the timely help.

I express my sincere appreciation for the excellent typing by R.N. Srivastava.

- Prashant Vijay Waknis

ABSTRACT

Various adaptive robot control algorithms have been studied in search of an algorithm that is suitable for implementation using low-cost microprocessors. It is desired that the controller tracks a desired time-based trajectory as closely as possible over a wide range of manipulator motion and payloads.

First, some adaptive control algorithms have been studied with respect to their tracking capability. Some basic works in the robot control area have been reviewed, a few new algorithms have been suggested and their critiques presented. In this work it is shown that, with simplified model, it is very difficult to achieve a 'guaranteed' control that can operate over wide range of motion and payloads.

A new discrete dynamic model has been derived for representation of robot. A computer simulation study has been conducted to evaluate the performance of the proposed model. The model shows good parameter convergence if there is no error in measurement.

A similar model has been used in Computed Torque Adaptive Control Patnaik, 1987 . Comparison of the robot control strategies shows that the Computed Torque Adaptive Control is most robust algorithm in terms of stability. This algorithm has been recommended for implementation. A dedicated computer architecture has been proposed for it and the algorithm has been suitably restructured for parallel processing.

CONTENTS

CHAPTER 1

INTRODUCTION

Industrial robots have evolved with the pressing need for increased productivity and delivery of end products of uniform quality. They are general purpose computer controlled manipulators playing important role in automation of industries.

To accomplish a desired high precision task, one would like to serve the manipulator's joint actuators such that the manipulator follows a desired path. The control problem of a manipulator can be divided into two coherent subproblems - the trajectory planning and the motion control. The motion control problem consists of obtaining dynamic models of the manipulator and using these models to determine control laws or strategies to achieve desired performance. The problem can also be referred to as 'trajectory-tracking problem'.

1.1 CONVENTIONAL MOTION CONTROL:

The motion control can be classified into two major categories:

(1) Joint motion control

(2) Resolved motion control (Cartesian space control).

1.1.1 JOINT MOTION CONTROL:

The desired motion is specified by a time-based trajectory of the manipulator in joint coordinates. The control task is to track the trajectory so that the errors between the desired and actual joint positions are minimum.

Most of the work in robot control area is based on joint motion control. A table loop-up method has been suggested by Albus, 1975 . It requires tremendous computer memory. It has been shown Hollerbach, 1980 that computing input generalized forces using recursive Newton-Euler equations is most efficient, but fast computer is needed to obtain satisfactory sampling frequency and further fast computation should use several CPUs Luh, Lin, 1982 . The use of variable structure theory has been proposed in Young, 1978 . This method is difficult to implement since exact determination of switching instances for the control input is difficult. Linear optimal control has been proposed in Kahn, Roth, 1971 , Takegaki, Arimoto, 1981 , and Vukobratovic, 1983 . However, linearization over whole motion range is difficult and the errors caused due to linearization are difficult to compensate for. The computed torque technique has been discussed in Luh et.al., 1980 , Luh, 1983 and Asada, Slotine, 1985 . They uses a structure identical to manipulator dynamics for generating the control inputs, and hence explicitly accounts for the nonlinear system dynamics. Implementation of such a control scheme needs the dynamic parameters of the system. Since very accurate measurement of these parameters is not possible, a controller based on this method will not be able to minimize the tracking error due to parameter uncertainties and unknown loads.

1.1.2 RESOLVED MOTION CONTROL:

Here, the desired motion is specified by a time-based trajectory of the manipulator in Cartesian space coordinates

(world coordinates). The control is applied to the joints so as to achieve the desired end-effector motion in world coordinates.

The resolved motion schemes are proposed in Whitney, 1969 , Luh et.al., 1980 and Paul, Wu, 1982.

1.1.3 DIFFICULTIES IN CONVENTIONAL APPROACH:

These control algorithms are somewhat inadequate because they require accurate modeling of the arm dynamics and neglect the changes of the load in a task cycle. The changes in the payload often are significant enough to render these strategies ineffective. The result is reduced servo response speed and damping, which limits the precision and speed of the end effector. Any significant gain in performance for tracking the desired trajectory as closely as possible over a wide range of manipulator motion and payloads require the consideration of adaptive control techniques.

1.2 ADAPTIVE ROBOT CONTROL - BASIC REVIEW:

In general, an adaptive controller can be defined as a controller which has some way of generating informations regarding the system dynamics and then utilizing them for control purpose. Usually an adaptive controller employs the past sequence of input and output data to gain knowledge about the system parameters.

Recently, various adaptive control algorithms have been proposed. Dubowsky, DesForges, 1979 devised a Model Reference Adaptive Control (MRAC) law using the steepest descent method for a class of manipulators with counterbalances in order to

handle nonlinearities and payload. But they neglected coupling between joints of the manipulator and did not assure the convergence of their control law. Takegaki, Arimoto, 1981 developed MRAC which consists of a feedforward control to reduce the effects of the gravity force and a feedback control to compensate for position errors, speed errors and other disturbances. The control is based on Lyapunov direct method which ensures convergence of the control law under the assumption of slowly time-varying dynamics of the manipulator. But, they assumed low speed motion to neglect Coriolis and centrifugal forces.

Various self-tuning control strategies have been proposed which fit the input-output data of the system with an autoregressive model. Leininger, Wang, 1982 used pole-placement technique, Koivo, Guo, 1983 used multivariable model based on LQG control approach, Liu, 1985 combined LQG control with pole-placement. Koivo, 1985 and Leininger et.al., 1986 extended their respective algorithms for resolved motion control. All these methods neglect the coupling forces between the joints which may be severe for manipulators with rotary joints.

Adaptive Perturbation Control theory has been proposed by Lee, Chung, 1984 . This strategy may be more appropriate for various manipulators because it takes all the interaction forces between the joints into consideration. The Adaptive Perturbation Control is characterized by a feedforward component, which is the nominal torque, values, and a feedback component which is computed separately and simultaneously in parallel. The Adaptive Perturbation Control theory has also been used by Choi et.al., 1985 . Their adaptive regulation scheme is

devised based on the Lyapunov direct method, which generates variational control that regulates the perturbation in the vicinity of the desired trajectory. These methods have to compute nominal torques and these torques cannot be computed exactly. The adaptive perturbation control is also extended to work as resolved-motion control Lee and Lee, 1984 .

Horowitz, Tomizuka, 1986 combined MRAC and PID control techniques. The design method presented by them is based on the hyperstability approach. The coupling among the joints and nonlinear terms are explicitly considered.

Patnaik, 1987 developed a new approach. Basec Computed Torque Technique is utilized to build the adaptive controller. The robot model assumed is nonlinear and represents the robot arm as closely as possible.

1.3 THE PROBLEMS ASSOCIATED WITH MOTION CONTROL:

Robot arm is highly nonlinear and multivariable system. There are interactions among the joints. Various models - linear and nonlinear, single-input-single-output and multi-input-multi-output, have been proposed to achieve robot control.

Most of the current industrial approaches to robot control problem treat each joint of the robot arm as a simple joint servomechanism. The servomechanism approach models the varying dynamics of a manipulator inadequately because it neglects the motion and configuration of the whole arm mechanism. The changes in the parameters of the robot sometimes are significant and they lead to degradation of performance in terms of

speed and tracking accuracy or even failure of these simple control strategies. On the other hand, significant performance gain in robot arm control require the consideration of more efficient dynamic models, sophisticated control approaches and the use of dedicated computer architectures and parallel processing techniques.

1.4 OBJECTIVE OF THE THESIS:

The objective of the present work is to study various adaptive control algorithms, in particular, study possibility of a simpler algorithm that would control the robot arm so as to track the desired trajectory as closely as possible over a wide range manipulator motion and payloads. It is also desired to propose a dedicated computer architecture and parallel processing technique that would do fast computations needed by a sophisticated control algorithm based on an efficient dynamic model.

1.5 BREAKDOWN OF CHAPTERS:

In Chapter 2, various adaptive control algorithms are presented. Various cases are considered to highlight their advantages and disadvantages. Their performance is observed with computer simulation.

Some basic works done in the adaptive robot control area are reviewed in Chapter 3. Some new algorithms are then proposed and their critiques are presented. Convergence and stability issues of the adaptive robot control algorithms are discussed.

In Chapter 4, firstly Computed Torque Adaptive Control technique is reviewed. A new model is derived which can be utilized for robot control. The effects of measurement noise and effects of neglecting some system dynamic parameters from the model are also investigated.

Chapter 5 provides implementation details of multi-processing architecture. The Computed Torque Adaptive Control algorithm is considered for implementation.

Chapter 6 enlists the various conclusions drawn through discussion and simulation studies and leaves a note on the scope for further work.

CHAPTER 2

SELF-TUNING AND ADAPTIVE CONTROL OF SISO MODEL

This chapter presents study of some commonly used self-tuning control algorithms. These algorithms or their variations are used for the robot control problem described in Chapter 3.

2.1 SELF-TUNING CONTROL:

In practice, a very common situation is the control of systems whose parameters are unknown. It thus becomes necessary to go through the steps of plant experiments, parameter estimation, computation of control strategies and implementation. This procedure is quite time consuming and tedious.

From a practical point of view, it is thus meaningful to consider control systems with unknown parameters. The plant parameters may be constant or time varying. For these systems, it is reasonable to look for strategies that will converge to the optimal or near optimal performance that could be derived if the system characteristics were known. Such algorithms are called self-tuning or self-adjusting strategies. When the characteristics of the process (system) are changing, the word 'adaptive' is used.

The regulators considered are described by the block diagram of Figure 2.1. The regulator can be thought of as being composed of three parts: a parameter estimator, a controller and a third part which relates the controller parameters to the estimated parameters.

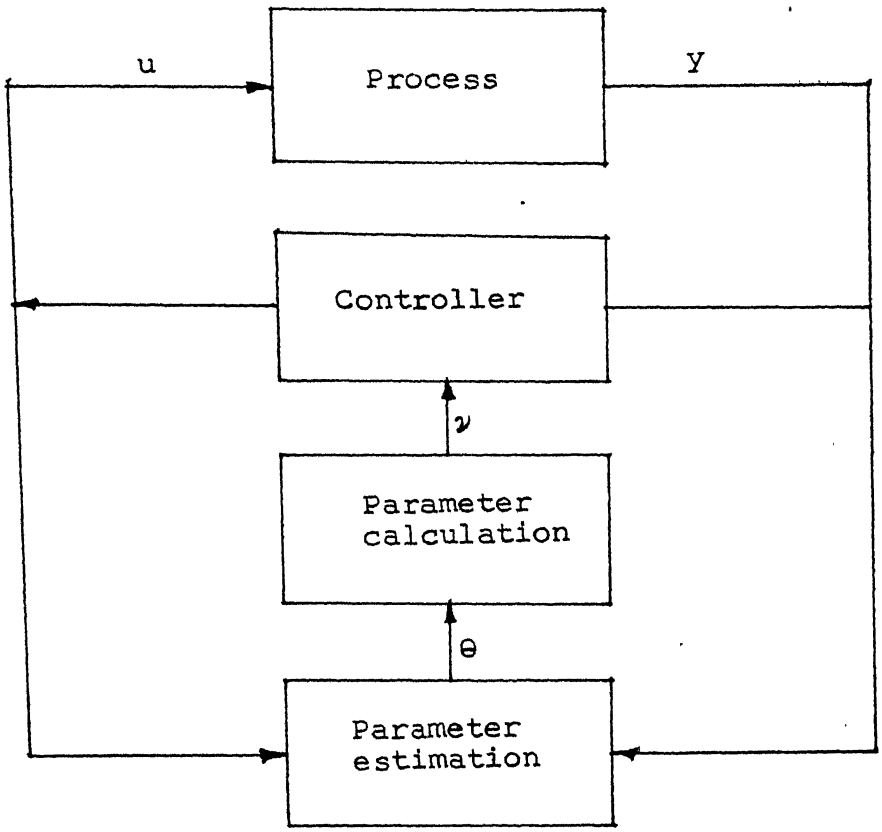


Fig. 2.1: Scheme of self-tuning system.

It is assumed that the process to be controlled can be described by the stochastic difference equation

$$A(z^{-1})\ y(k) = z^{-d}\ B(z^{-1})\ u(k-1) + C(z^{-1})\ e(k) \qquad (2.1)$$

where u and y are scalar input and output signals and $\{e(k)\}$ a disturbance which is a sequence of independent random variables with zero mean, and d is the delay associated with the control input. $A(z^{-1})$, $B(z^{-1})$ and $C(z^{-1})$ are polynomials in the backward shift operator z^{-1} defined by

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{n_a} z^{-n_a}$$

$$B(z^{-1}) = b_o + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{n_b} z^{-n_b}$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{n_c} z^{-n_c}$$

It is to be noted that the polynomials $A(z^{-1})$ and $C(z^{-1})$ start with 1 (coefficient of z^{0} is 1) and are called monic polynomials.

2.2 SYSTEM IDENTIFICATION:

The first step in self-tuning control algorithm is to estimate the plant transfer function based on input and output data. Two schemes are usually used in practice. They are presented below.

2.2.1 RECURSIVE LEAST SQUARES (RLS):

Assume that the system can be modeled by the following equation.

$$A(z^{-1})\, y(k) = z^{-d}\, B(z^{-1})\, u(k-1) + V(k) \qquad (2.2)$$

where $V(k)$ is a sequence of zero-mean white noise.

The following recursive equations then are used to identify the system parameters Ljung, Soderstrom, 1983 .

$$\underline{\theta}(k) = \underline{\theta}(k-1) + \underline{L}(k)\, y(k) - \underline{\theta}^{T}(k-1)\, \underline{\Psi}(k) \qquad (2.3)$$

$$\underline{L}(k) = \frac{P(k-1)\, \underline{\Psi}(k)}{\lambda + \underline{\Psi}^{T}(k)\, P(k-1)\, \underline{\Psi}(k)} \qquad (2.4)$$

$$P(k) = \frac{1}{\lambda}\, P(k-1) - \underline{L}(k)\, \underline{\Psi}^{T}(k)\, P(k-1) \qquad (2.5)$$

where

$\underline{\theta}$ is the parameter vector given by

$$\underline{\theta}^{T} = \hat{a}_1 \quad \hat{a}_2 \ldots \hat{a}_{n_a} \quad \hat{b}_o \ldots \hat{b}_{n_b} \qquad (2.6)$$

$\hat{a}_i$, $i = 1,\ldots,n_a$ and $\hat{b}_i$, $i = 0,\ldots,n_b$ being the estimated counterparts of a_i, $i = 1,\ldots,n_a$ and b_i, $i = 0,\ldots,n_b$, respectively and

$\underline{\psi}$ is a regression vector given by

$$\underline{\psi}^T(k) = -y(k-1) - y(k-2) - \ldots - y(k-n), \; u(k-d-1), - \ldots - u(k-d-m-1) \qquad (2.7)$$

and λ is a scalar forgetting factor used to discount old data. The range of λ is between 0 and 1. To start the algorithm, the following initial conditions are used

$$P(0) = I, \quad \text{large}$$

$$\underline{\theta}(0) = \underline{0}$$

The term $y(k) - \underline{\theta}(k-1)\,\underline{\psi}(k)$ is referred to as 'estimation error' or 'filtering residual'.

2.2.2 RECURSIVE EXTENDED LEAST SQUARES (RELS):

If $\nu(k)$ in eqn. (2.2) is a moving average process given by

$$\nu(k) = C(z^{-1})\, e(k)$$

$$= (1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{n_c} z^{-n_c})\, e(k)$$

then the process equation becomes the same as given in eqn. (2.1). In this case, the coefficients of polynomial C need to be estimated along with those of A and B.

Define

$$\underline{\theta}^T = \hat{a}_1 \ldots \hat{a}_{n_a} \; \hat{b}_0 \ldots \hat{b}_{n_b} \; \hat{c}_1 \ldots \hat{c}_{n_c} \qquad (2.8)$$

and

$$\Psi^T(k) = -y(k-1) - \ldots - y(k-n)\ u(k-d-1) \ldots u(k-d-m-1)\ \hat{e}(k-1) \ldots \hat{e}(k-n_c) \quad (2.9)$$

where

$\hat{e}(j)$ is the estimated value of $e(j)$.

Then same set of equations (2.3)-(2.5) are used to estimate the system parameters.

2.3 MINIMUM VARIANCE (MV) CONTROL:

In this section, Minimum Variance (MV) strategy is considered. The structure of the regulator is first described. Then its extension to servoproblem is discussed.

2.3.1 M.V. REGULATOR:

Let the system to be controlled be given by eqn. (2.1). Then in M.V. control a cost function of the form

$$V_1 = E\ y^2(k) \quad (2.10)$$

or

$$V_2 = \lim_N E\ \frac{1}{N} \sum_1^N y^2(k) \quad (2.11)$$

is minimized to obtain the control input u(k) Astrom, 1970 . Let the control be computed from

$$u(k) = -\frac{G(z^{-1})}{F(z^{-1})}\ y(k) \quad (2.12)$$

where,

$$F(z^{-1}) = B(z^{-1})\ F_1(z^{-1}) \quad (2.13)$$

Furthermore, $G(z^{-1})$, $F_1(z^{-1})$ are solutions of the Diophantaine equation

$$C(z^{-1}) = A(z^{-1})\, F_1(z^{-1}) + z^{-d-1}\, G(z^{-1}) \qquad (2.14)$$

and $F_1(z^{-1})$ and $G(z^{-1})$ are polynomials in z^{-1} given by

$$F_1(z^{-1}) = 1 + f_1 z^{-1} + \ldots + f_d z^{-n_f}$$

$$G(z^{-1}) = g_0 + g_1 z^{-1} + \ldots + g_{n_g} z^{-n_g}$$

The orders of polynomial G and F_1 are $n_g = n_a - 1$ and $n_c \leq n_a + d$. If $d = 0$, then the solutions of the Diophantine equation becomes

$$F_1(z^{-1}) = 1 \quad \text{and} \quad z^{-1}\, G(z^{-1}) = C(z^{-1}) - A(z^{-1})$$

and the control law is simply given by

$$u(k) = -\frac{C(z^{-1}) - A(z^{-1})}{z^{-1}\, B(z^{-1})}\, y(k) \qquad (2.15)$$

The self-tuning control can be obtained in the following two ways.

Algorithm 1:

Step 1: Assuming that the system is governed by eqn. (2.1), the parameters are identified using RELS method.

Step 2: Eqns. (2.13) and (2.14) are solved with the estimated parameters to obtain the polynomials $G(z^{-1})$ and $F(z^{-1})$.

Step 3: Control is computed using eqn. (2.12).

Steps 1, 2 and 3 are repeated for each sampling period.

Algorithm 2:

Step 1: The parameters are identified assuming that the system is governed by eqn. (2.2).

Step 2: Equations (2.13) and (2.14) are solved with $C(z^{-1}) = 1$ to obtain the polynomials $G(z^{-1})$ and $F(z^{-1})$.

Step 3: Control is computed using eqn. (2.12).

Steps 1, 2 and 3 are repeated for each sampling period.

The RLS method is used in Algorithm 2. As a result, the estimated parameters can be biased if the polynomial C is not 1. However, it has been shown that if the parameters converge to some steady-state values, not necessarily to the actual ones, the control law obtained is in fact the minimum variance control law Astrom, Wittenmark, 1973 . This is the self-tuning property. Thus, even if the process is governed by eqn. (2.1) and the control law is derived based on RLS scheme eqns. (2.3), (2.4), (2.5) , the closed-loop system exhibits minimum variance property.

The following conditions are necessary for successful application of the minimum variance control.

(1) The polynomial B has all zeros inside the unit circle, i.e., system is minimum phase.

(2) The polynomial C has all zeros inside the unit circle.

(3) Time delay of the system is known.

It must be noted that if the system is non-minimum phase, the control strategy will still be minimum variance strategy. This strategy will, however, be so sensitive that the slightest variance will create an unstable closed-loop system.

variation

2.3.2 THE COMBINED SERVO AND REGULATOR PROBLEM:

A self-tuner which can both eliminate disturbances and follow a command input can also be obtained using the basic minimum variance principle.

Consider the system given in eqn. (2.1). Let $y_r(k)$ be a reference signal. Introducing a reference control signal $u_r(k)$ through

$$A(z^{-1})\ y_r(k) = B(z^{-1})\ u_r(k - d - 1) \tag{2.16}$$

and subtracting this equation from eqn. (2.1) the following equation is obtained

$$A(z^{-1})\ y(k) - y_r(k) = B(z^{-1})\ u(k - d - 1) - u_r(k - d - 1) + C(z^{-1})\ e(k) \tag{2.17}$$

The M.V. controller for (2.17) is then given by,

$$u(k) - u_r(k) = -\frac{G(z^{-1})}{F(z^{-1})}\ y(k) - y_r(k) \tag{2.18}$$

where F and G are solutions of eqns. (2.13) and (2.14).

Eliminating u_r from eqn. (2.18), the control law can be written as

$$u(k) = -\frac{G(z^{-1})}{F(z^{-1})}\ y(k) - y_r(k) + \frac{A(z^{-1})}{B(z^{-1})}\ y_r(k + d + 1) \tag{2.19}$$

or when using the eqns. (2.13) and (2.14),

$$u(k) = -\frac{G(z^{-1})}{F(z^{-1})}\ y(k) + \frac{C(z^{-1})}{F(z^{-1})}\ y_r(k + d + 1) \tag{2.20}$$

The control law thus consists of two parts: a feedback from the output error (2.19) or the output (2.20) and a feedom the reference value. This is illustrated in Figure

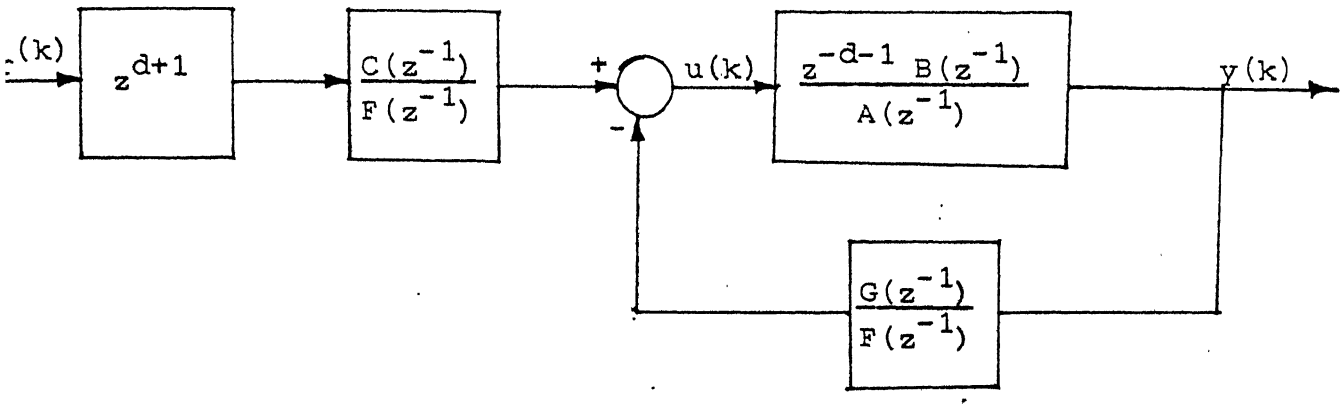


Fig. 2.2: M.V. Servo Tracker.

The closed loop system then is

$$y(k) = \frac{F(z^{-1})}{B(z^{-1})}\, e(k) + y_r(k) \qquad (2.21)$$

where $\frac{F(z^{-1})}{B(z^{-1})}$ is a polynomial of order d. Thus, the controller minimizes the effect of disturbance and at the same time the system follows the reference value.

2.4 POLYNOMIAL LINEAR QUADRATIC GAUSSIAN (LQG) REGULATOR:

In this section, a polynomial approach is considered for LQG regulator. This approach is much simplified as compared to the state variable LQG controller whose structure is considerably more complex. In state variable LQG controller, the design calculations which are done in each step involve the solution of a Riccati equation or, equivalently, a spectral factorization [Astrom, Wittenmark, 1980].

2.4.1 LQG PROBLEM FORMULATION:

Let the system be described by eqn. (2.1). Then the following quadratic performance index is selected.

$$J(k) = P(z^{-1})\, y(k + d') - Q(z^{-1})\, y_r(k)^2 + R(z^{-1})\, u(k)^2 \quad (2.22)$$

$P(z^{-1})$, $Q(z^{-1})$ and $R(z^{-1})$ being polynomials in the backward shift operator z^{-1} given by

$$P(z^{-1}) = 1 + p_1 z^{-1} + \ldots + p_{n_p} z^{-n_p}$$

$$Q(z^{-1}) = q_o + q_1 z^{-1} + \ldots + q_{n_q} z^{-n_q}$$

$$R(z^{-1}) = r_o + r_1 z^{-1} + \ldots + r_{n_r} z^{-n_r}$$

and $d' = d + 1$

The cost function can be selected to specify a wide range of designs that result in desirable closed-loop behavior for different applications. For example, by choosing $P(z^{-1}) = Q(z^{-1}) = 1$ and $R(z^{-1}) = r_o$ a polynomial tracker can be obtained. The controlled output will then follow a delayed reference signal $y_r(k)$. This is desirable in robot control applications.

2.4.2 THE CONTROL STRATEGY:

The optimal LQG control law is defined by the recursion

$$P(z^{-1})\, \hat{y}(k + d'/k) + \frac{r_o}{b_o} R(z^{-1})\, u(k) - Q(z^{-1})\, y_r(k) = 0 \quad (2.23)$$

where

$$\hat{y}(k + j/k) = \frac{G_j(z^{-1})}{C(z^{-1})} + \frac{B(z^{-1})\, F_j(z^{-1})}{C(z^{-1})} u(k - d' + j)$$

$$0 < j \leq d' \quad (2.24)$$

$F_j(z^{-1})$ and $G_j(z^{-1})$ are polynomials of degree j-1 and n_a-1 respectively. These polynomials are obtained as solutions of the Diophantine equations

$$A(z^{-1})\ F_j(z^{-1}) + z^{-j} G_j(z^{-1}) = C(z^{-1}) \qquad (2.25)$$

for $j = 1, 2, \ldots, d'$.

A more explicit recursion for u(k) is found as follows Lewis, 1986 .

Using the definition of $A(z^{-1})$ eqn. (2.23) is rewritten as

$$\sum_{j=o}^{n_p} P_j\ y(k + d' - j/k) + \frac{r_o}{b_o} R(z^{-1})\ u(k) - Q(z^{-1})\ y_r(k) = 0 \qquad (2.26)$$

Defining new polynomials

$$G'(z^{-1}) = \sum_{j=o}^{n_p} P_j\ G_{d'-j}(z^{-1}) \qquad (2.27)$$

$$F'(z^{-1}) = \sum_{j=o}^{n_p} P_j\ B(z^{-1})\ F_{d'-j}(z^{-1})\ z^{-j} + \frac{r_o}{b_o} C(z^{-1})\ R(z^{-1}) \qquad (2.28)$$

Eqn. (2.26) becomes

$$F'(z^{-1})\ u(k) = - G'(z^{-1})\ y(k) + C(z^{-1})\ Q(z^{-1})\ y_r(k) \qquad (2.29)$$

The polynomial LQG controller given by eqn. (2.29) can be represented by the block diagram shown in Figure 2.3.

2.4.3 THE ALGORITHM:

The stepwise algorithm is as follows.

Data : Polynomials $P(z^{-1})$, $Q(z^{-1})$, $R(z^{-1})$ and system delay d.

Step 1: Identify the system parameters using RELS equations.

Step 2: Solve (2.25) for F_j and G_j, $0 < j \leq d'$.

Step 3: Solve eqn. (2.24) for $\hat{y}(k + j/k)$ for $0 < j \leq d'$.

Step 4: Use eqn. (2.23) to find control u(k).

The steps 1, 2, 3 and 4 are repeated for each sampling period.

2.4.4 REMARKS:

(1) This algorithm, though simpler than the state space formulation, requires a good deal of computation. The number of computations increase with the degree of polynomials in the system, the delay d and the cost function.

(2) Though the algorithm provides a wide flexibility in the choice of cost functions, one has to verify that the closed-loop system is stable. This obviously restricts choice of cost functions.

2.5 SELF-TUNING CONTROLLER BASED ON POLE-ZERO PLACEMENT:

M.V. control suffers if the plant is non-minimum phase. The problem here is that M.V. strategies try to cancel the non-minimum phase zeros with unstable poles of the controller. Hence, M.V. regulators are highly sensitive to parameter fluctuations.

Another practical difficulty which occurs in M.V. controllers, is that large amplitude control signals are generated which can exceed the control limits. The hard limiters are introduced in such cases and this degrades the regulator response.

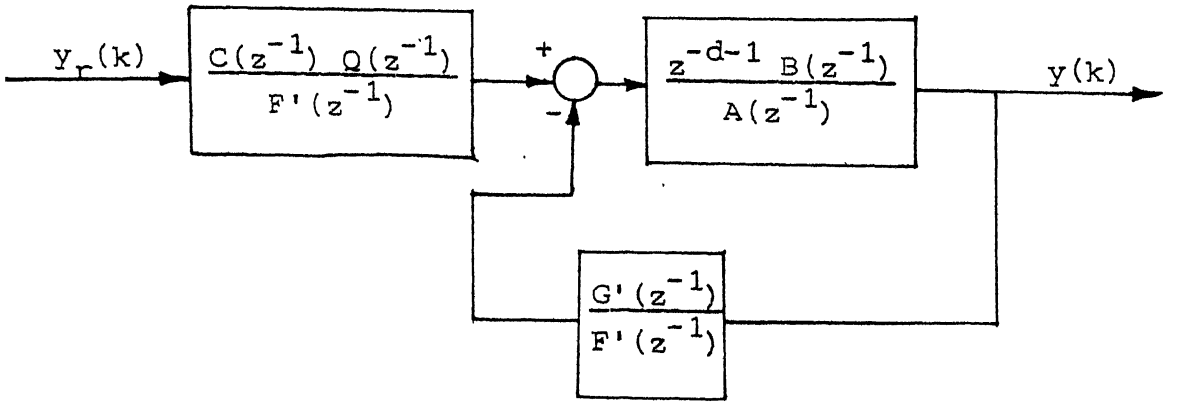


Fig. 2.3: Polynomial LQG Controller.

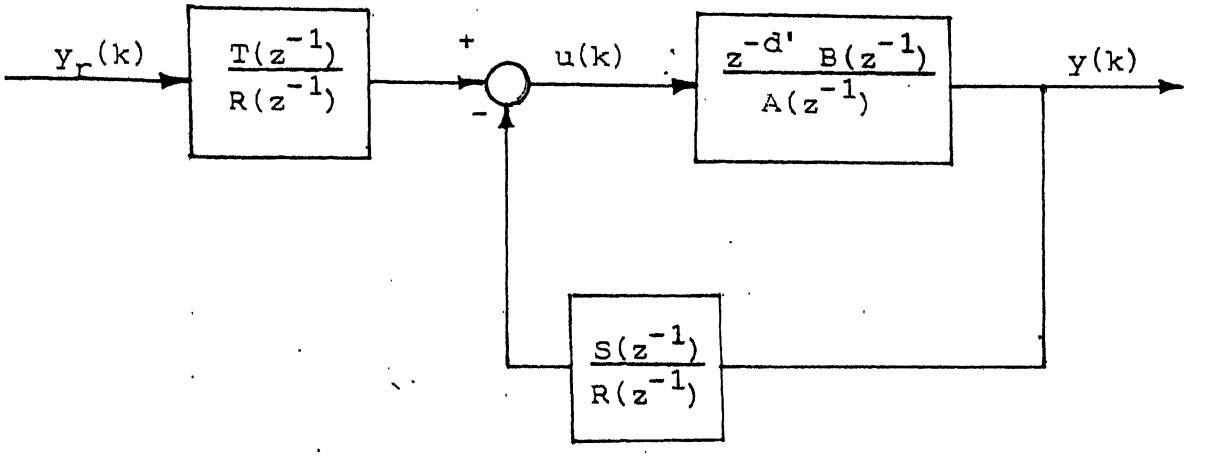


Fig. 2.4: Pole Zero Placement Design.

Further, M.V. controllers need to know the system time delay 'a priori'.

These difficulties can, however, be eliminated by use of the LQG control strategy. The self-tuning regulator based on the LQG formulation has the drawback of being more complicated than the M.V. self-tuning regulator. The reason for this is that too many design calculations are involved in each step. The self-tuners based on pole-zero placement are much simpler to use, while at the same time less sensitive to parameter fluctuations.

2.5.1 PROBLEM FORMULATION AND DESIGN CONSIDERATION:

Let the system be characterized by eqn. (2.1).

The transfer function of the system is then

$$G(z^{-1}) = z^{-d'} \frac{B(z^{-1})}{A(z^{-1})} \quad \text{where} \quad d' = d + 1 \tag{2.30}$$

It is desired that the transfer function from command input y_r to the output is given by

$$G_m(z^{-1}) = z^{-d'} \frac{B_m(z^{-1})}{A_m(z^{-1})} \tag{2.31}$$

such that the closed-loop system is stable.

DESIGN CONSIDERATIONS:

Let the general regulator be described by

$$R(z^{-1})\, u(k) = T(z^{-1})\, y_r(k) - S(z^{-1})\, y(k) \tag{2.32}$$

The closed-loop transfer function relating y to y_r is then given by,

$$\frac{z^{-d'} TB}{AR + z^{-d'} BS} = \frac{z^{-d'} B_m}{A_m} \qquad (2.33)$$

Here, the backward shift operator is omitted for brevity.

The design problem is thus equivalent to the algebraic problem of finding polynomials R, S and T such that eqn. (2.33) holds. It follows from eqn. (2.33) that the factors of B which are not also the factors of B_m must be factors of R. The polynomial B can be factorized as

$$B = B^+B^- \qquad (2.34)$$

where all the zeros of B^+ are inside the unit circle and all the zeros of B^- are outside the unit circle.

A necessary condition for solvability of the servo problem is thus that the specifications are such that

$$B_m = B_{m_1} B^- \qquad (2.35)$$

Since degree of A_m is normally less than that of $AR + z^{-d'} BS$, it is clear that there are factors in eqn. (2.31) which cancel. This polynomial which cancels in theeright hand side of eqn. (2.31) can be interpreted as the observer polynomial A_o Astrom, Wittenmark, 1979[30].

The block diagram of the closed-loop system is shown in Figure 2.4. The controller can be interpreted as being composed of a feedforward path with the transfer function T/R and a feedback wpath with a transfer function -S/R.

2.5.2 THE ALGORITHMS:

Algorithm 1:

Data : The polynomials A_m and A_o, both with zeros inside unit circle, and B_{m_1} are given.

Step 1: Estimate the parameters of the model by least squares.

Step 2: Factor the polynomial B into B^+ and B^-.

Step 3: Solve the linear equation

$$AR_1 + z^{-d'} B^- S = A_m A_o$$

where, S and R_1 are polynomials in z^{-1} with deg S = deg A-1 and deg R_1 = deg A_m + deg A_o - deg A.

Step 4: Calculate the control signal from $Ru = Ty_r - Sy$ with $R = R_1 B^+$ and $T = A_o B_{m_1}$.

The steps 1, 2, 3 and 4 are repeated at each sampling period.

A special case of Algorithm 1 is obtained when all process zeros are well damped. It is then reasonable to have a pole-placement design where all the process zeros are cancelled with controller poles. Under this hypothesis, the pole-placement procedure can be simplified. The corresponding self-tuning algorithm then becomes.

Algorithm 2:

Step 1: Estimate the parameters of the model by least squares.

Step 2: Determine the polynomials R_1 and S such that

$$AR_1 + z^{-d'} S = A_m A_o.$$

Step 3: Use the control law

$$BR_1 u = Ty_r - Sy$$

where

$$T = A_o B_m$$

The steps 1, 2 and 3 are repeated for each sampling period.

Since the controller poles here are same as the process zeros, this algorithm is not suitable for non-minimum phase systems.

2.5.3 POLE SHIFTING CONTROLLER:

This is a special case of pole placement controller. Here, closed-loop poles are determined from the open-loop poles by the relation,

$$A_m(z^{-1}) = A(\omega z^{-1})$$

$$= 1 + \omega a_1 z^{-1} + \omega^2 a_2 z^{-2} + \ldots + \omega^{n_a} a_{n_a} z^{-n_a}$$

where $0 < \omega < 1$

The choice of ω will shift the open loop-poles towards more stable locations in closed loop. It is seen in Figure 2.5.

The advantage here is that only one parameter ω is needed to specify closeness of the closed-loop poles to the origin Ghosh et.al., 1984 .

2.6 SELF-TUNING PID CONTROLLER:

In this section Proportional-plus-derivative-plus-integral (PID) control using pole-placement technique is discussed.

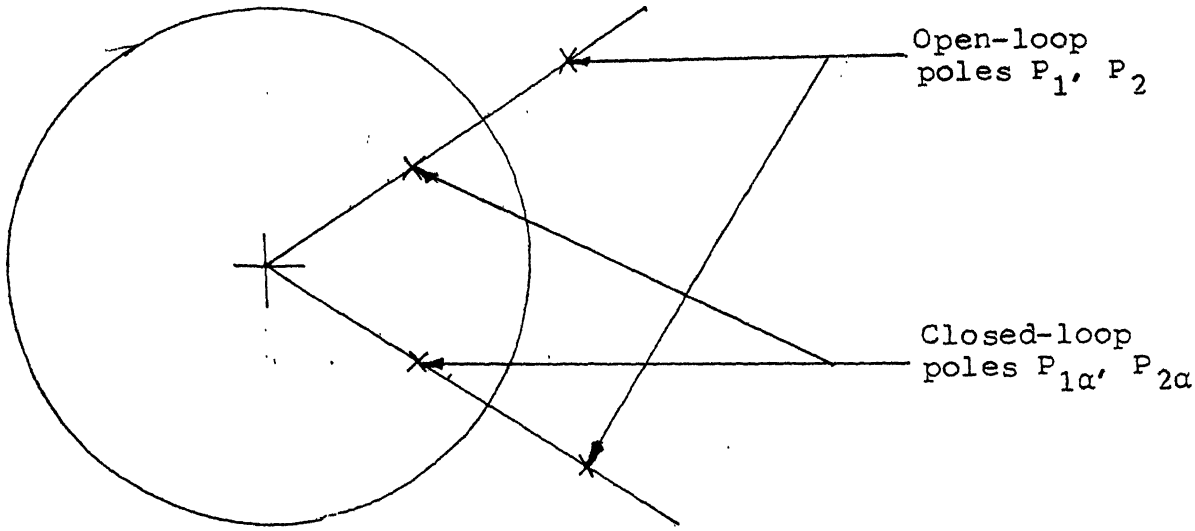


Fig. 2.5: Pole Shifting Controller.

2.6.1 DIGITAL PID CONTROLLER:

An analog PID controller, used as a compensator, is depicted in Figure 2.6. The PID control algorithm in such a case has a general form

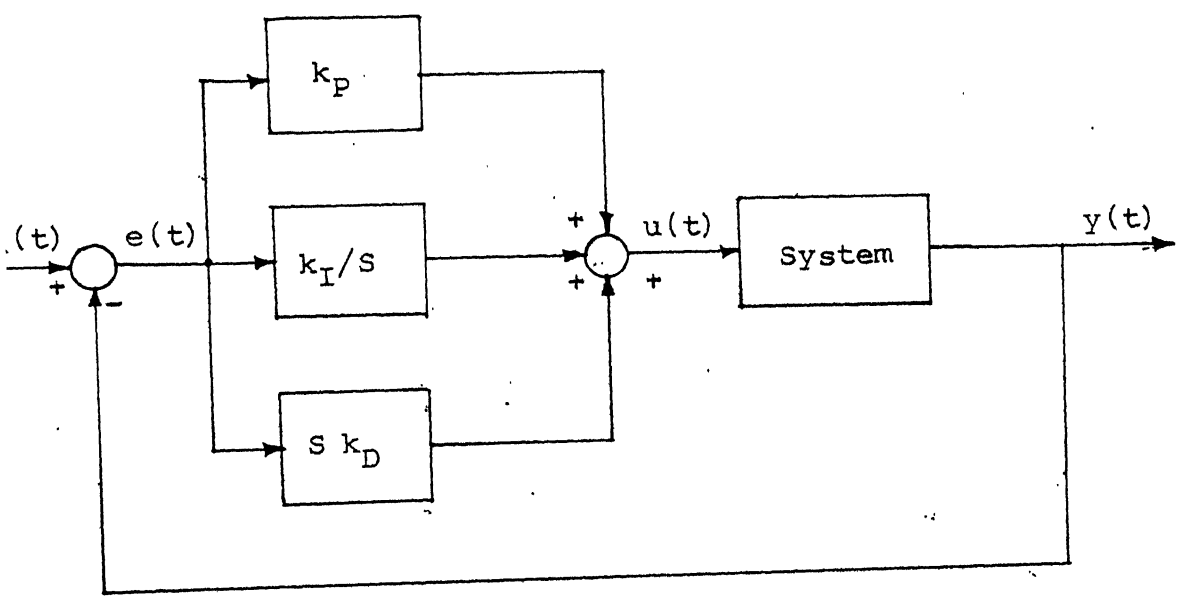


Fig. 2.6: Analog PID Controller.

$$\frac{U(s)}{E(s)} = k_P + \frac{k_I}{s} + s\,k_D \qquad (2.36)$$

Discretizing eqn. (2.36) and putting it in the incremental form, the following equation is obtained Isermann, ~~1980~~ 1981.

$$u(k) = u(k-1) + (S_1 + S_2 z^{-1} + S_3 z^{-2})\, e(k) \qquad (2.37)$$

where,

$$S_1 = k_P + k_I T + \frac{k_D}{T}$$

$$S_2 = -k_P - \frac{2k_D}{T}$$

$$S_3 = \frac{k_D}{T}$$

and T is the sampling interval.

If set point signal $y_r(k)$ is included only in integral term, a PID structure is obtained which avoids "set point and derivative kick". That is, a sudden change in the output after fast change in the set point is reduced Isermann, 1981.

The structure so obtained is,

$$u(k) = u(k-1) + a\, e(k) - \beta(z^{-1})\, y(k) \qquad (2.38)$$

where

$$\beta(z^{-1}) = \beta_0 (1 - z^{-1})\left(1 - \frac{\beta_2}{\beta_0} z^{-1}\right)$$

$$a = T k_I$$

$$\beta_0 = k_P + \frac{k_D}{T}$$

$$\beta_2 = \frac{k_D}{T}$$

2.6.2 SELF-TUNING PID CONTROLLER:

Let the SISO system to be controlled be given by

$$A(z^{-1})\ y(k) = z^{-1}\ B(z^{-1})\ u(k) + A(z^{-1})\ \eta(k) \qquad (2.39)$$

where η is the measurement noise

and the PID controller structure chosen be

$$R(z^{-1})\ u(k) = \alpha\ e(k) - \beta(z^{-1})\ y(k) \qquad (2.40)$$

where,

$$R(z^{-1}) = (1 - z^{-1})(1 + \gamma\ z^{-1}) \qquad (2.41)$$

The block diagram of the controller is then shown in Figure 2.7.

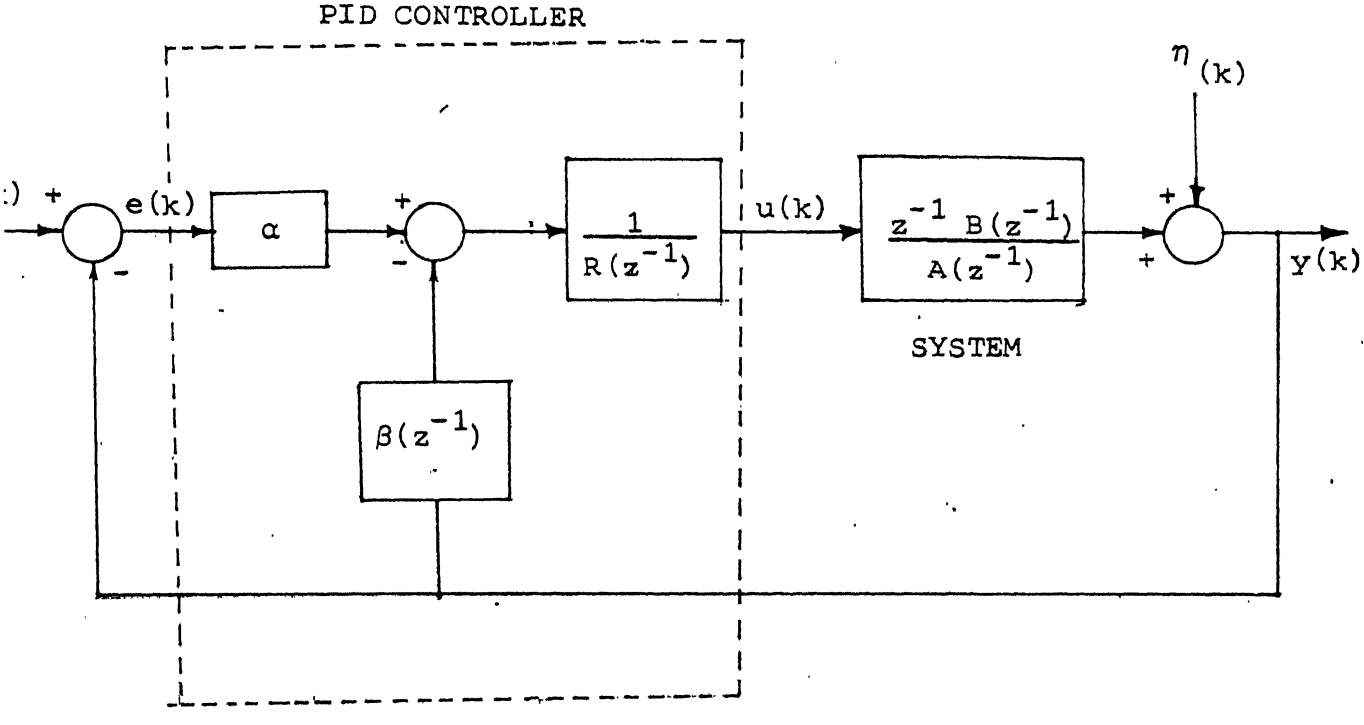


Fig. 2.7: Self-tuning PID Controller.

From Figure 2.7, it is seen that the closed-loop transfer function is given by,

$$\frac{y(k)}{y_r(k)} = \frac{z^{-1} \alpha B(z^{-1})}{A(z^{-1}) R(z^{-1}) + z^{-1} B(z^{-1}) \alpha + \beta(z^{-1})}$$

Let the desired closed-loop transfer function be $\frac{z^{-1}}{A_m}$, where $A_m(z^{-1})$ is the desired characteristic polynomial.

Equating the closed-loop transfer function with the desired one and manipulating further, the following equation is obtained,

$$\frac{B(z^{-1})}{A(z^{-1})} = \frac{\frac{1}{\alpha}(1 + r z^{-1})}{\frac{A_m(z^{-1}) - z^{-1}}{1 - z^{-1}} - \frac{\beta_0}{\alpha} z^{-1} + \frac{\beta_2}{\alpha} z^{-2}} \qquad (2.42)$$

Looking at eqn. (2.42) it can be argued that instead of a parameter vector containing a_i's and b_i's the following parameter vector can also be used

$$\underline{\theta}^T = \frac{\beta_0}{\alpha}, \frac{\beta_2}{\alpha}, \frac{1}{\alpha}, \frac{r}{\alpha}$$

If the RLS equations are now used for identification, the controller parameters are directly obtained. It is to be noted that the two-step procedure, that is, parameter estimation and control law design has been reduced to single one by suitable plant reparametrization. Algorithms of this type are called 'algorithms based on implicit identification'.

If $A_m(z^{-1}) = 1$, the controller exhibits deadbeat response. In this case, the estimation error is created directly comparing y(k) with its predicted value. Otherwise, it must be filtered by $\frac{A_m(z^{-1}) - z^{-1}}{1 - z^{-1}}$. The filtering is done as shown in Figure 2.8. Note that $G(z^{-1}) = \frac{A_m(z^{-1}) - z^{-1}}{1 - z^{-1}}$

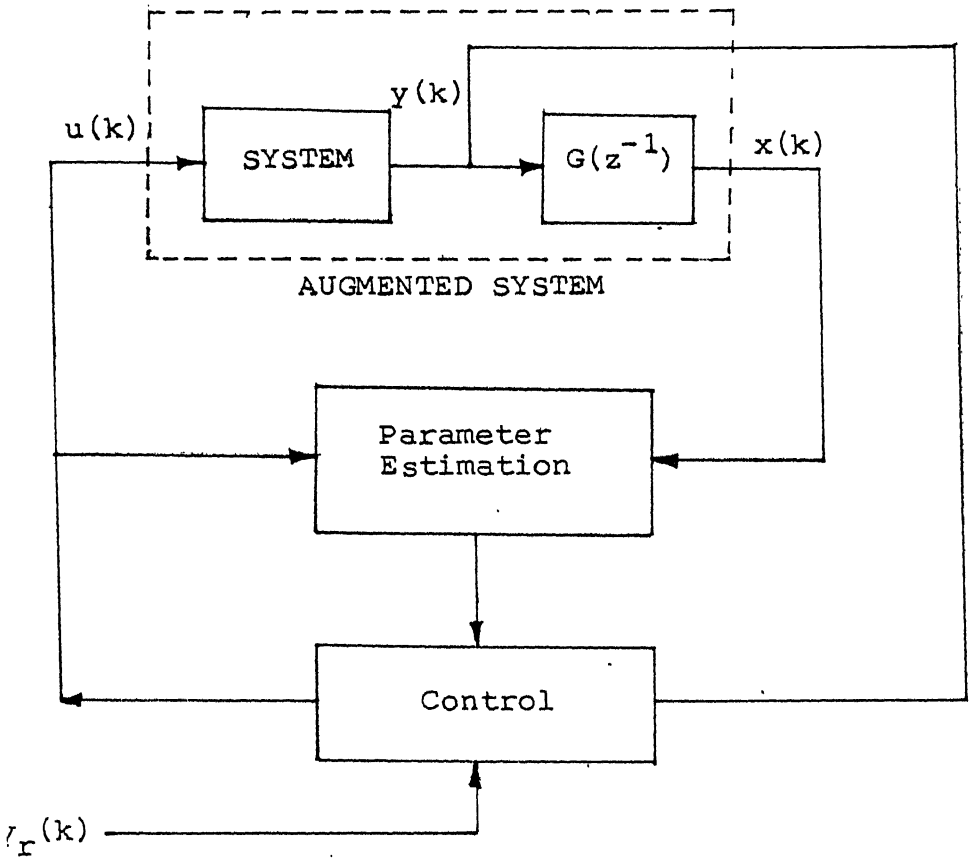


Fig. 2.8: Filtering.

System to be controlled is augmented by $G(z^{-1})$. Parameter estimation is done for this system. Thus, the regression vector is

$$\underline{\psi}^T(k) = [x(k-1), \; -x(k-2), \; u(k-1), \; u(k-2)]$$

where $x(k)$ is calculated from

$$x(k) = G(z^{-1})\, y(k).$$

The control is desired such that $y(k)$ follows the reference $y_r(k)$.

2.6.3 THE ALGORITHM:

Data : The polynomial $A_m(z^{-1})$.

Step 1: $G(z^{-1})$ is calculated from given $A_m(z^{-1})$.

Step 2: $x(k)$ is computed from measurement of $y(k)$.

Step 3: Parameter estimation is done for the augmented system using RLS method.

Step 4: Control is computed using eqn. (2.40).

Steps 2, 3 and 4 are repeated for each sample.

REMARK: Since, all process zeros are cancelled by the controller, this scheme fails for non-minimum phase systems.

2.7 SIMULATION STUDIES:

Performance of the control strategies mentioned in the previous section has been studied through digital computer simulation. In particular, their reference tracking capability has been investigated.

For this purpose the P matrix in the system identification has been initialized to αI where $\alpha = 1000.0$. All the elements in parameter vector $\underline{\theta}$ are initialized to zero. For simplicity, time invariant system equations have been selected for simulation studies. The forgetting factor λ is chosen as 1.0.

Noise of standard deviation 0.0015 has been introduced at the output.

To start the algorithm only the parameter vector θ is updated for the first few samples. During this period, a nominal control input has been applied to the system. The controller has been switched on after elements of parameter matrix are non zero. This is done to avoid overflow due to division by zero.

2.7.1 M.V. CONTROL:

The system to be controlled is represented by the following difference equation.

$$y(k) + 0.6\, y(k-1) + 0.11\, y(k-2) + 0.006\, y(k-3) = u(k-1) + 0.5\, u(k-2) + 0.05\, u(k-3)$$

Two cases have been studied.

In the first case, the output is expected to follow reference signal which toggles between two values. In the second case, the reference signal is a sinusoidal waveform.

Figure 2.9 shows that the output tracks the reference value very closely in both the cases.

2.7.2 LQG CONTROL:

The system to be controlled is represented by the following difference equation.

$$y(k) - 0.39\, y(k-2) + 0.07\, y(k-3) = u(k-1) + 0.5\, u(k-2) + 0.06\, u(k-3)$$

A sinusoidal input is used as a reference signal.

It has been discussed that the controller acts as a tracker if $P(z^{-1}) = Q(z^{-1}) = 1$. Three different values of $R(z^{-1})$ are considered.

(1) $R(z^{-1}) = 0.0$

(2) $R(z^{-1}) = 0.2$

(3) $R(z^{-1}) = 0.9$

It can be seen from Figure 2.10 that when there is no penalty on control, i.e. when $R(z^{-1}) = 0.0$, the LQG controller

acts as an A.V. controller. Tracking accuracy degrades on the penalty on control is increased.

2.7.3 POLE PLACEMENT CONTROL:

It has been discussed that the pole shifting method offers a convenient way to decide closeness of the closed-loop poles to origin. Only one parameter, α, is needed to be specified. Hence, this approach is used in simulation studies. Three cases have been considered.

(1) The system to be controlled is

$$y(k) + 2.4\, y(k-1) + 1.91\, y(k-2) + 0.504\, y(k-3)$$
$$= u(k-1) + 0.5\, u(k-2) + 0.06\, u(k-3)$$

The system is minimum phase. Hence Algorithm 2 is applied. Various polynomials in the algorithm are selected as follows.

$$A_o = 1 + 0.5\, z^{-1}$$
$$B_m = 1.62 - 0.59\, z^{-1}$$
and $$\alpha = 0.6$$

It can be seen from Figure 2.11 that the output of the system tracks the reference signal.

(2) The system to be controlled is

$$y(k) - 0.39\, y(k-2) + 0.07\, y(k-3)$$
$$= u(k-1) + 2.0\, u(k-2) + 0.75\, u(k-3)$$

This is a non-minimum phase system. Algorithm 1 is applied with the following data.

α = 0.6

A_o = $1 + 0.5\ z^{-1}$

B_{m_1} = $1 - 0.66\ z^{-1}$

It is evident from Figure 2.11 that the output tracks the reference signal in this case also.

(3) The system to be controlled is

$y(k) + 0.1\ y(k-1) - 3.12\ y(k-2) - 2.16\ y(k-3)$

$= u(k-1) + 2\ u(k-2) + 0.75\ u(k-3)$

The system is unstable as well as non-minimum phase. Algorithm 1 is applied with

α = 0.4

A_o = $1 + z^{-1} + 0.12\ z^{-2}$

B_{m_1} = $1 - 0.84\ z^{-1}$

The reference signal tracking is shown in Figure 2.11.

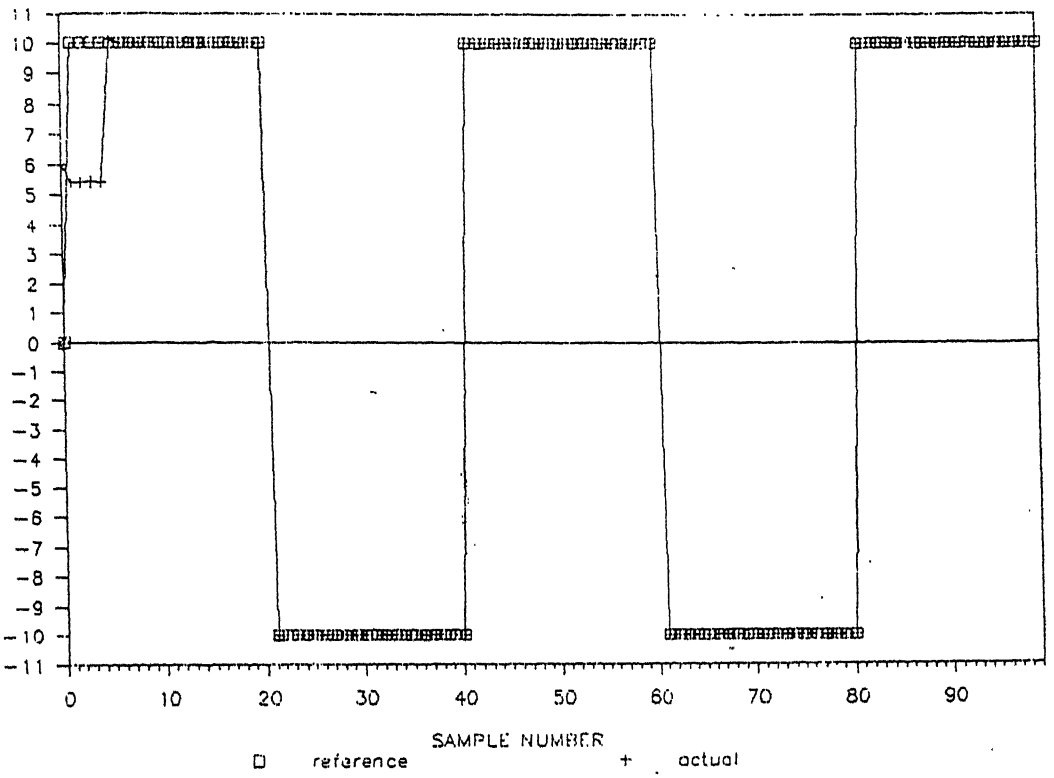
2.7.4 PID CONTROL:

The system to be controlled is

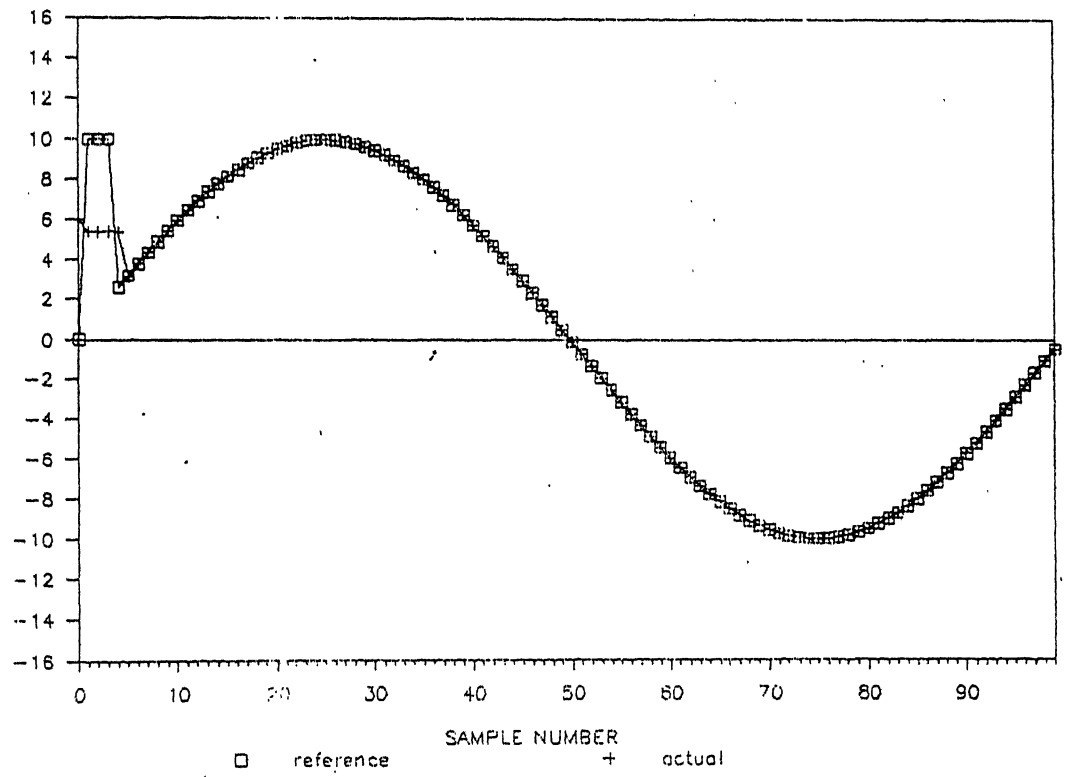
$y(k) - y(k-1) + 0.477\ y(k-2) = 0.9\ y(k-1) - 0.7\ u(k-2)$

The closed-loop poles are placed at $0.1 \pm j\ 0.435$.

The tracking performance for square wave and sinusoidal reference signals is shown in Figure 2.12.

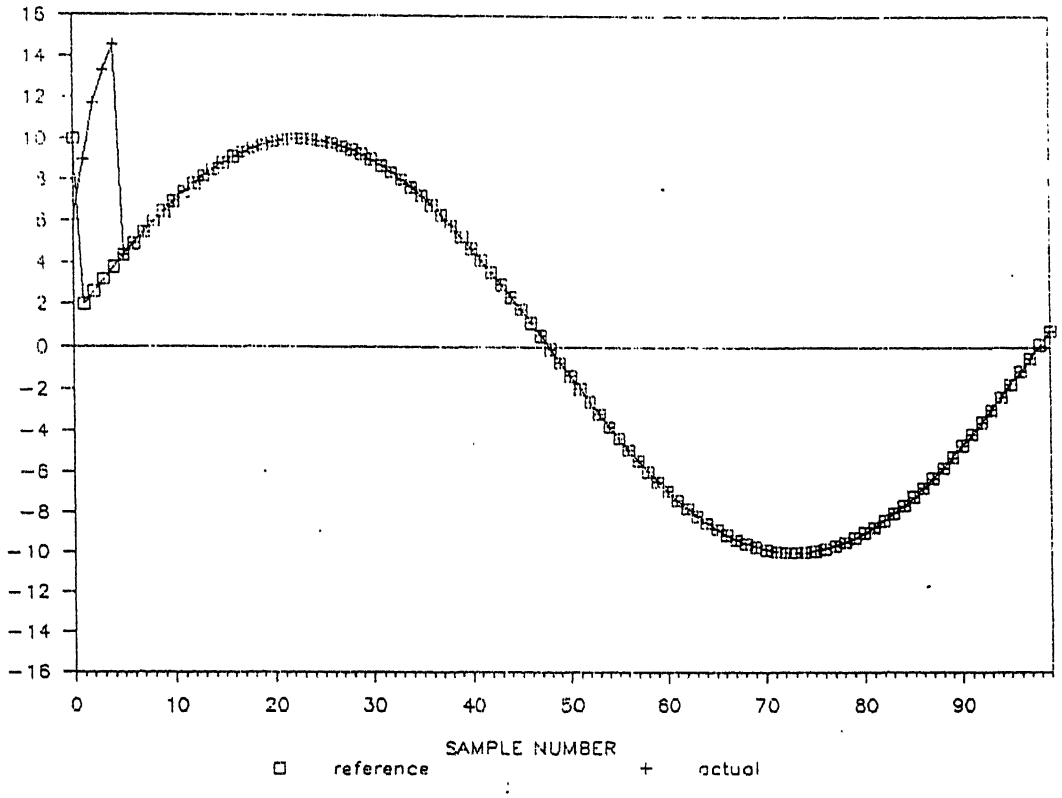


(a)

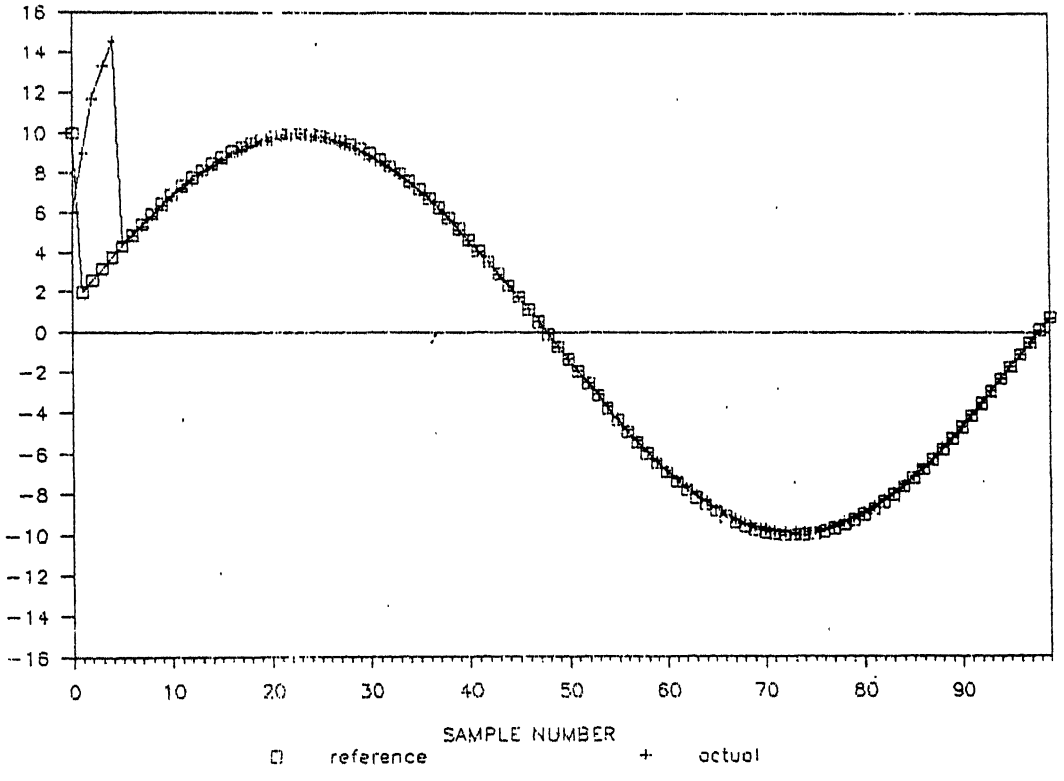


(b)

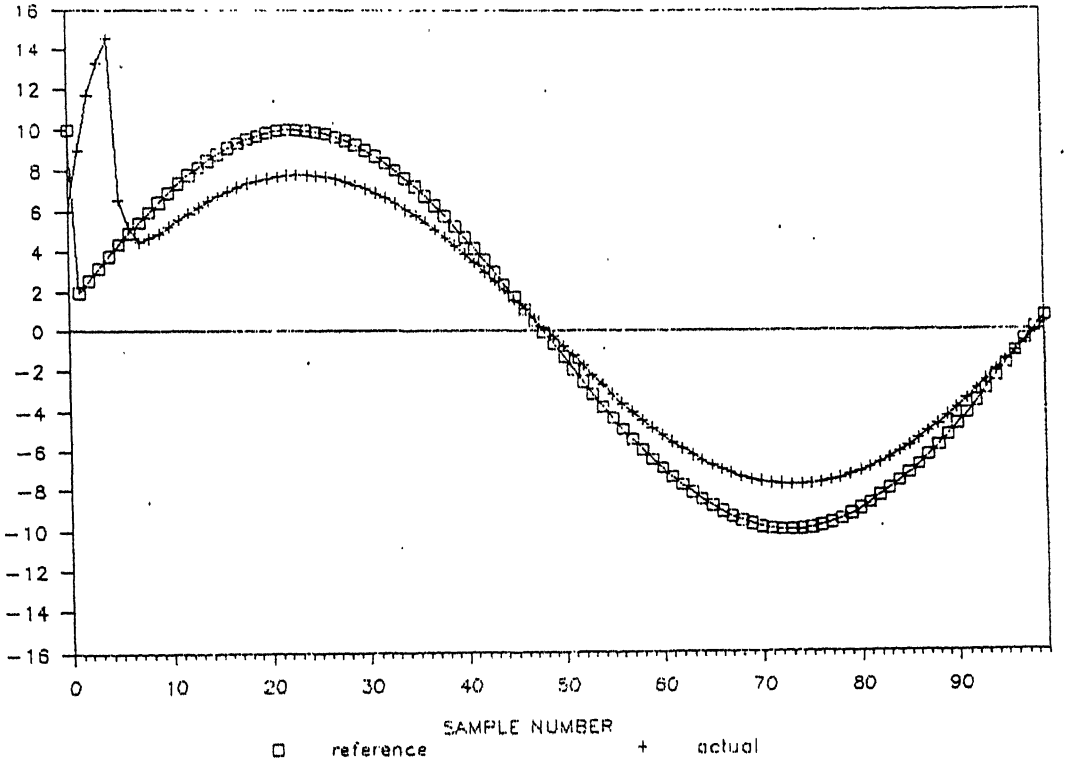
Fig. 2.9: M.V. Control (a) Square-wave Tracking
(b) Sine-wave Tracking


16
14
12
10
8
6
4
2
0
−2
−4
−6
−8
−10
−12
−14
−16
0
10
20
30
40
50
60
70
80
90
SAMPLE NUMBER
reference
actual

(a)


16
14
12
10
8
6
4
2
0
−2
−4
−6
−8
−10
−12
−14
−16
0
10
20
30
40
50
60
70
80
90
SAMPLE NUMBER
reference
actual

(b)



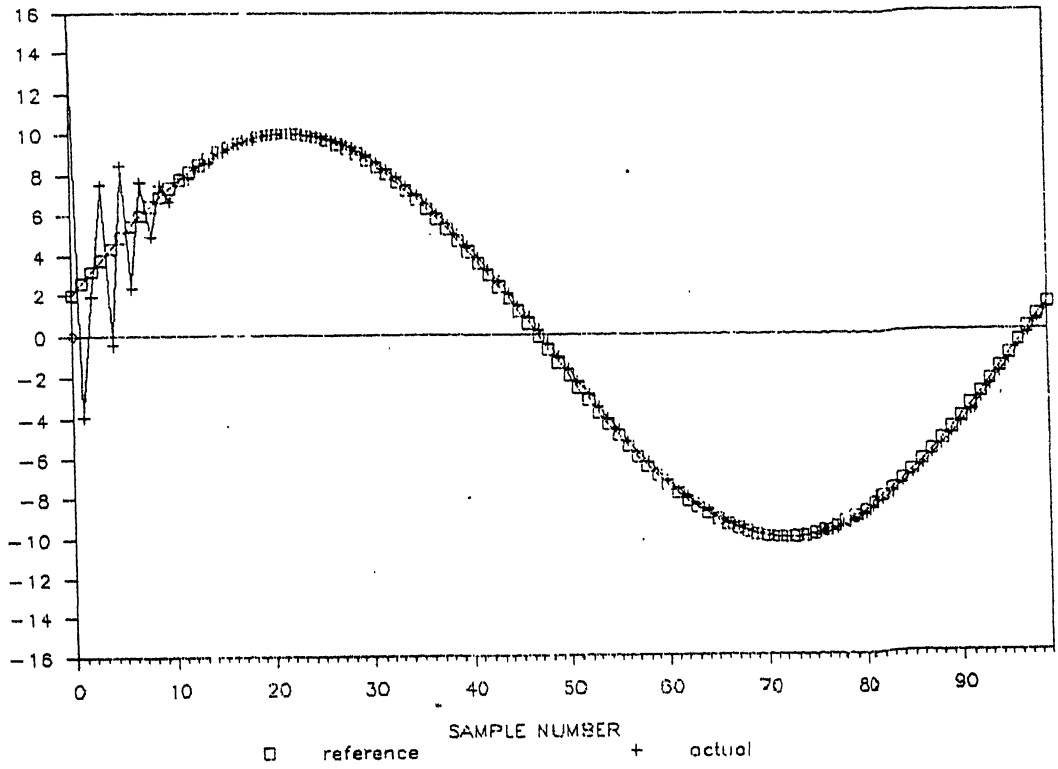
(c)

Fig. 2.10: LQG Control

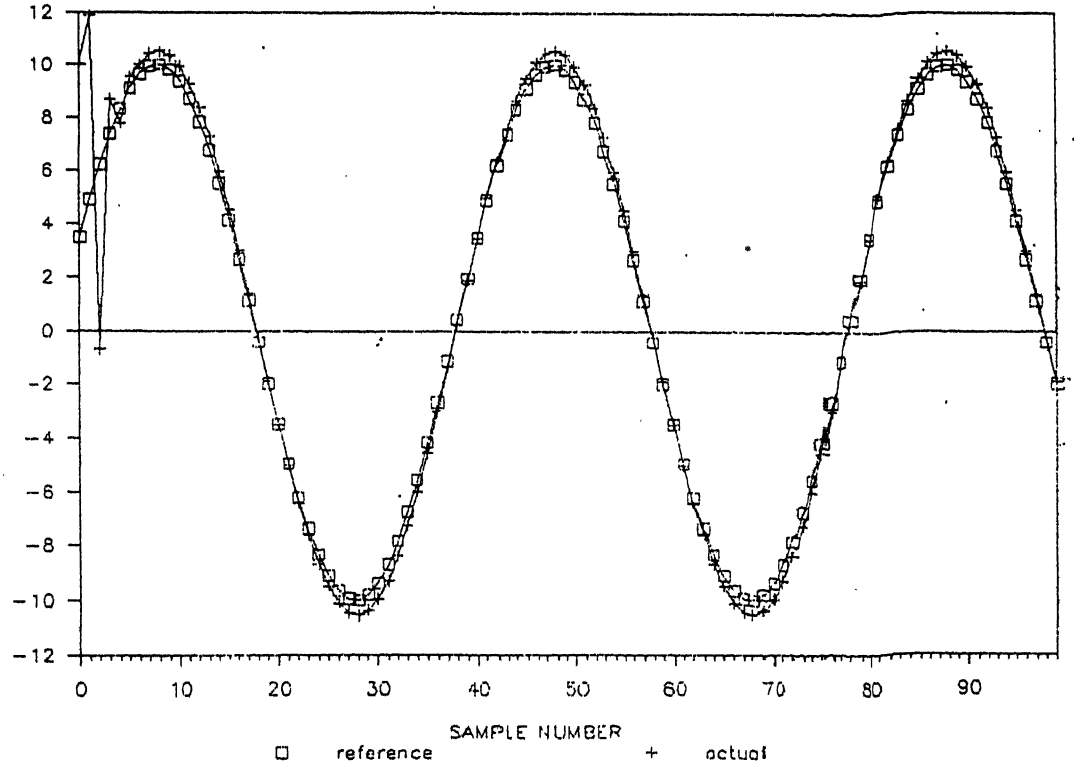
(a) $R(z^{-1}) = 0.0$

(b) $R(z^{-1}) = 0.2$

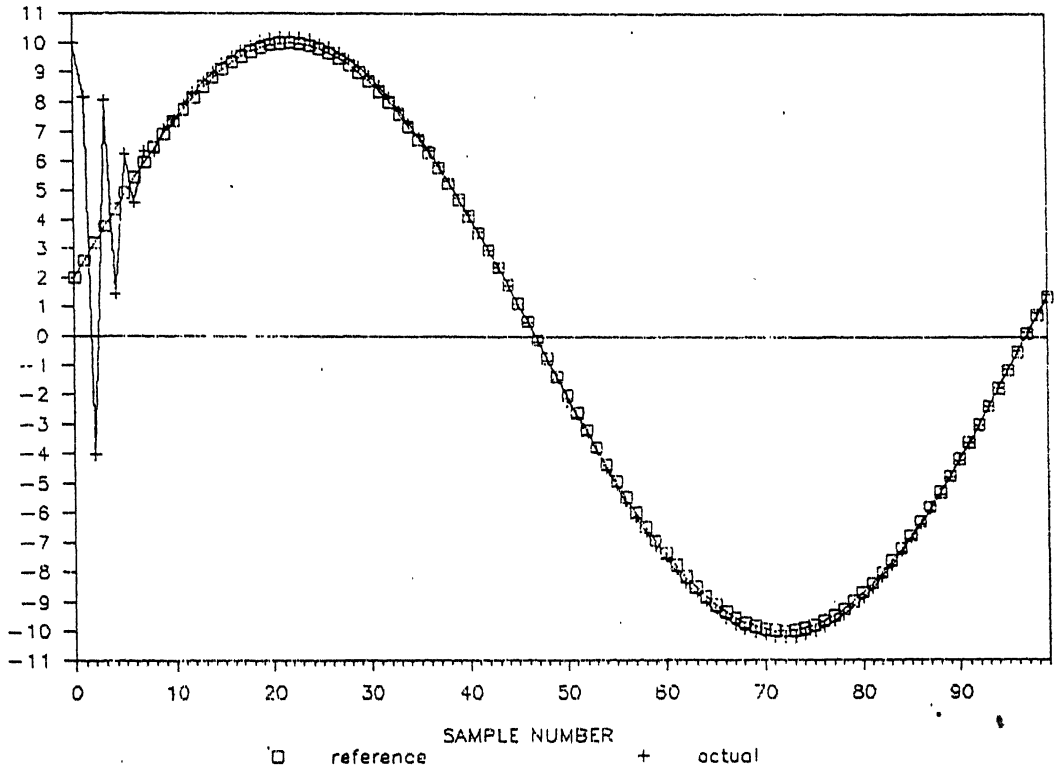
(c) $R(z^{-1}) = 0.9$


16
14
12
10
8
6
4
2
0
−2
−4
−6
−8
−10
−12
−14
−16
Y
0
10
20
30
40
50
60
70
80
90
SAMPLE NUMBER
reference
actual

(a)


12
10
8
6
4
2
0
−2
−4
−6
−8
−10
−12
Y
0
10
20
30
40
50
60
70
80
90
SAMPLE NUMBER
reference
actual

(b)



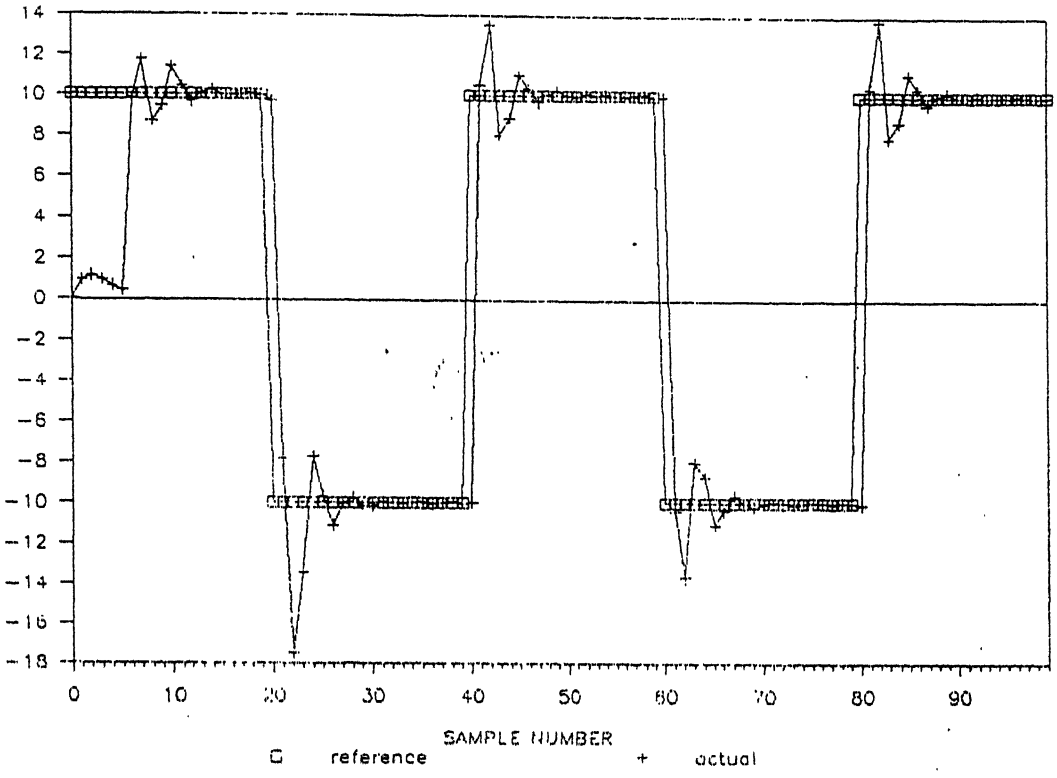
(c)

Fig. 2.11: Pole Placement Control

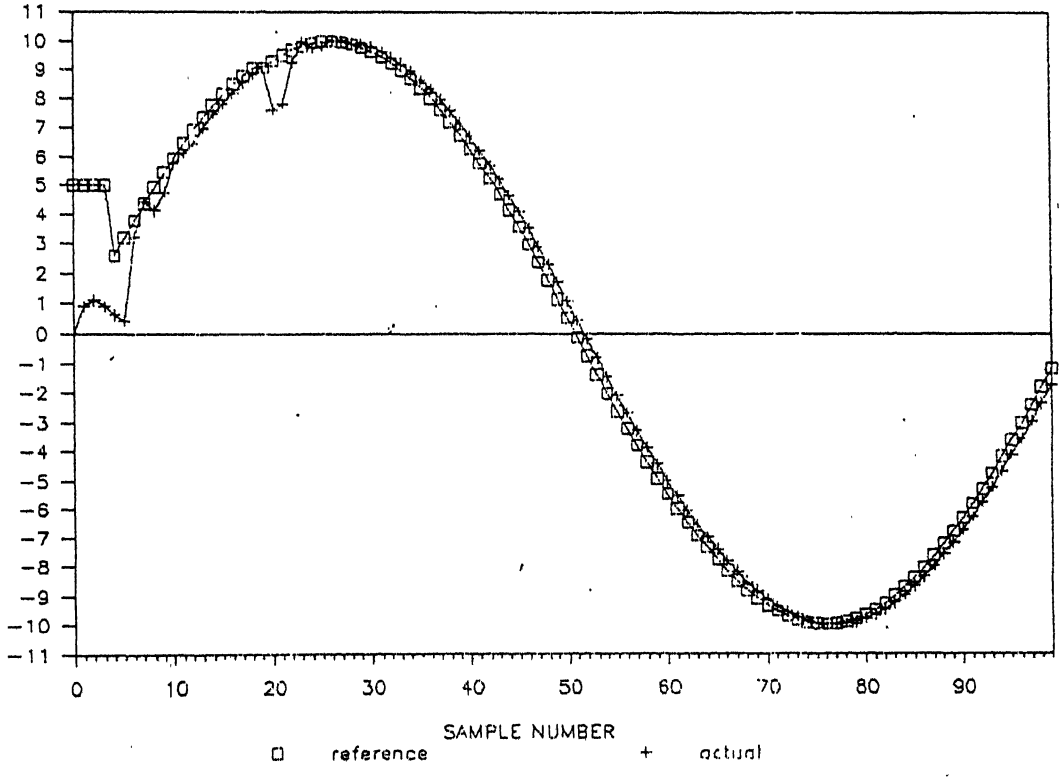
(a) Case 1: Stable, minimum phase system

(b) Case 2: Stable, non-minimum phase system

(c) Case 3: Unstable, non-minimum phase system.



(a)



(b)

Fig. 2.12: Adaptive PID Control
(a). Square-wave Tracking

CHAPTER 3

ADAPTIVE CONTROL OF MANIPULATORS BASED ON CENTRALIZED AND DECENTRALIZED MODELS

Chapter 2 presented a summary of commonly used control algorithms. Application of these algorithms to the robot control problem is the prime concern of this chapter.

The chapter is organized as follows. First the dynamic model of a manipulator is discussed. This is followed by some basic work carried out in this area. Next, some new algorithms are suggested and their critiques presented. Finally, convergence and stability issues of the robot control algorithms are discussed.

3.1 DYNAMIC MODEL OF THE MANIPULATOR:

'A priori' information needed for control is a set of differential equations describing the dynamic behavior of a manipulator. In general, the Lagrange-Euler equation of motion for an n-jointed manipulator can be expressed in vector matrix notation as Lee, Chung, 1982

$$D(q)\,\ddot{\underline{q}} + \underline{H}(q, \dot{q}) + \underline{G}(q) = \qquad (3.1)$$

where q, $\dot{q}$ and $\ddot{q}$ are n-dimensional vectors signifying the joint position, velocity and acceleration, respectively. $D(q)$ is an $(n \times n)$ symmetric matrix. It includes the acceleration related coefficients of the joints and the effects of link inertia. $\underline{H}(q, \dot{q})$ is an n-dimensional vector signifying Coriolis and

centrifugal torques. $\underline{G}(q)$ is an n-dimensional gravity vector. The n-dimensional vector $\underline{\tau}$ is the system input (torques/forces).

The dynamic equations of motion for a manipulator are highly coupled and non-linear. The elements of $D(q)$, $\underline{H}(q, \dot{q})$, $\underline{G}(q)$ contain trigonometric functions. Moreover, the parameter values vary with changes in positions and payloads.

3.2 ADAPTIVE CONTROL:

The non-adaptive control algorithms require accurate modeling of the arm dynamics and they neglect the changes of the load in a task cycle. These changes in the payload of the controlled system often are significant enough to render these control strategies ineffective. Any significant gain in performance for tracking the desired time-based trajectory over a wide range of manipulator motion and payloads require the consideration of adaptive control techniques.

The adaptive control algorithms are much more complex in numerical sense than non-adaptive laws. Also, it is difficult to prove the stability of the overall system Vukobratovic, et.al., 1985 . However, one has to resort to the adaptive control if robot works in unknown conditions.

Almost all parameter estimation algorithms have been based on linear models. On the other hand, robot model is a highly nonlinear system. A theory which is compatible with nonlinear robot system has been proposed Vukobratovic et.al., 1985 in which a nonlinear discrete model is used. This model typically consists of a very large number of parameters. For a six degree of freedom robot, there are more than two thousand

scalar parameters. Moreover, one cannot use least squares algorithm since it would yield biased estimates. Extended least squares has to be resorted to. Even though this model is likely to give superb control, it is just impossible to implement it using today's VLSI technology.

Hence, linear models which are approximations of the robot dynamic model are used in practice. With the linear models, the adaptive control strategies can be broadly classified into two groups.

The centralized control structure is one in which the plant to be controlled (the robot arm) is considered as a unique multivariable system in which all the joint interactions are taken into account. In decentralized approach to control synthesis, the robot is regarded as a set of decoupled subsystems each corresponding to a separate joint and the coupling among them is neglected.

Adaptive Perturbation Control Lee, Chung, 1984 , Computed Torque Adaptive Control Patnaik, 1987 are examples of centralized control, whereas pole-placement self-tuning control Leininger, Wang, 1982 , PID self-tuner proposed in this chapter are examples of decentralized control approach.

The robot control strategies based on these two approaches are discussed subsequently.

3.3 POLE PLACEMENT SELF-TUNING (PPST) CONTROL OF MANIPULATORS Leininger, Wang, 1982 :

In planning pole-placement strategy, the dynamic interactions among the joints are assumed negligible. Hence, each

joint is controlled independently. The mathematical model for each joint is assumed to be described by

$$A(z^{-1})\ y(k) = B(z^{-1})\ u(k - d) + h(k - d) \qquad (3.2)$$

where

y is output (position) of the joint

u is the control input (torque)

h contains unmodelled effects such as gravity force and joint coupled interactions.

It is assumed to be constant for a joint. The control input, u(k), is calculated from

$$G(z^{-1})\ u(k) = y_r(k + d) - F(z^{-1})\ y(k) \qquad (3.3)$$

Substituting (3.3) into (3.2), the closed loop equation is obtained as

$$A(z^{-1})\ G(z^{-1}) + z^{-d}\ B(z^{-1})\ F(z^{-1})\ y(k) = B(z^{-1}) y_r(k) \qquad (3.4)$$

The constant h(t - k) is cancelled by incorporating it into the applied control. In assigning the closed-loop places, the parameter $F(z^{-1})$ and $G(z^{-1})$ can be obtained by solving the following system characteristic equation.

$$A(z^{-1})\ G(z^{-1}) + z^{-d}\ B(z^{-1})\ F(z^{-1}) = A_m(z^{-1}) \qquad (3.5)$$

where $A_m(z^{-1})$ is given by,

$$A_m(z^{-1}) = (1 + p_1 z^{-1})(1 + p_2 z^{-2}) \ldots (1 + p_m z^{-1}) \qquad (3.6)$$

where p_i, $i = 1,\ldots,m$ are desired closed-loop poles. The polynomials $A(z^{-1})$ and $B(z^{-1})$ used in eqn. (3.5) are first estimate

This algorithm is tested on a six joint JPL-Stanford manipulator.

3.4 ADAPTIVE LINEAR CONTROLLER (ALC) FOR ROBOT MANIPULATORS Koivo, Guo, 1983 :

The model of eqn. (3.1) is first linearized and then discretized by Euler's method. A multivariable discrete-time model is obtained in the form

$$\underline{y}(k) = \underline{a}_o + A_1\underline{y}(k-1) + A_2\underline{y}(k-2) + B_1\underline{u}(k-1) + B_2\underline{u}(k-2) + \underline{e}(k) \qquad (3.7)$$

where A_i, B_i are (n x n) matrices, n is the number of joints.

$\underline{y}(k)$ is an n-dimensional position vector (output)

$\underline{u}(k)$ is an n-dimensional torque vector (input)

$\underline{a}_o$ is an n-dimensional vector.

3.4.1 AN AUTOREGRESSIVE MODEL FOR MANIPULATOR MOTION:

The model obtained in eqn. (3.7) can be generalized as a multivariable ~~given by,~~ difference equation given by,

$$\underline{y}(k) = A(z^{-1})\,\underline{y}(k) + B(z^{-1})\,\underline{u}(k-d) + h + \underline{e}(k) \qquad (3.8)$$

where $A(z^{-1})$ and $B(z^{-1})$ are polynomial matrices defined by

$$A(z^{-1}) = A_1z^{-1} + A_2z^{-2} + \ldots + A_{n_a}z^{-n_a} \qquad (3.9)$$

$$B(z^{-1}) = B_o + B_1z^{-1} + \ldots + B_{n_a-1}z^{-(n_a-1)} \qquad (3.10)$$

Note that A_i, $i = 1,\ldots,n_a$ and B_i, $i = 0,\ldots,n_a-1$ are (n x n) matrices, and,

$\underline{h}$ is a vector which includes the effects of gravitational forces.

With this model, the parameters can be estimated using multivariable least squares algorithm.

3.4.2 SELF-TUNING ADAPTIVE CONTROLLER:

The performance criterion for the system is chosen as

$$I^k(\underline{u}) = E \left\| \underline{y}(k+d) - \underline{y}_r(k+d) \right\|_Q^2 + \left\| \underline{u}(k) \right\|_R^2 / \Psi(k-1) \qquad (3.11$$

where Ψ is the regression vector and $\|\cdot\|_R$ indicates the norm with weight R i.e. $\|\underline{u}\|_R^2 = \underline{u}^T R \underline{u}$. R is a positive semidefinite symmetric matrix, Q is a positive definite symmetric matri and $\underline{y}_r(.)$ describes the desired path vector as a sequence of discrete points. The expectation operation is conditioned on the available measurements upto and including time k-1.

The problem is to minimize the performance criterion (3.11) while satisfying the following constraint equation

$$\underline{y}(k) = \theta^T(k)\, \underline{\Psi}(k-1) + \underline{e}(k) \qquad (3.1$$

The control which minimizes (3.11) is determined by th following three equations:

$$R\,\underline{u}(k) + B_0^T Q\, \hat{\underline{y}}(k+d/k-1) - \underline{y}_r(k+d) = \underline{0} \qquad (3.1$$

$$\hat{\underline{y}}(k+r/k-1) = \theta^T(k-1)\, \hat{\underline{\Psi}}(k-1+r/k-1) \qquad (3.1$$

$$\hat{\underline{\Psi}}(k-1+r/k-1) = \hat{\underline{y}}^T(k-1+r/k-1), \ldots, \hat{\underline{y}}^T(k/k-1), \underline{y}^T(k-1), \ldots, \underline{y}^T(k+r-n); \underline{u}^T(k-d+r), \ldots, \underline{u}^T(k-d-n+1+r); 1^T \qquad (3.1$$

where $r = 0, 1, \ldots, d$

Θ is the parameter matrix defined by

$$\Theta = A_1 \vdots A_2 \vdots \cdots \vdots A_{n_a} \vdots B_o \vdots \cdots \vdots B_{n_a-1} \vdots h^T$$

The control algorithm has been tried using simulation of a JPL-Stanford manipulator for the following two cases.

(1) Separate Joint Control:

It is assumed that interactions between the joints are small. The coupling terms in the autoregressive model of the manipulator are omitted and each joint is controlled independent of the other.

(2) Adaptive Control for Interacting Joints:

The problem is solved using multivariable autoregressive model (3.8). In order to restrict complexity of this model, the variables of the model are decomposed into two groups on the basis of the mechanical structure of the manipulator. The coupling terms between the variables of the two groups are neglected

For example, the matrices A_j in eqn. (3.9) are written as

$$A_j = \begin{matrix} A_{j1} \\ \hline \\ A_{j2} \end{matrix}$$

where $j = 1, \ldots, n_a$ and A_{j1} and A_{j2} are both $(\frac{n_a}{2} \times \frac{n_a}{2})$ matrices.

3.5 CENTRALIZED ADAPTIVE PERTURBATION CONTROL (CAPC) Lee, Chung, 1984 :

Based on perturbation theory, Lee, Chung, 1984 (), proposed an adaptive control strategy which tracks a desired trajectory over a wide range of manipulator motions and payloads. Adaptive perturbation control takes all the interactions among the various joints into consideration. The control is based on linearized perturbation equation in the vicinity of a nominal trajectory.

3.5.1 PERTURBATION EQUATION OF MOTION:

Defining the state output vector as $\underline{x} = \underline{q}^T, \dot{\underline{q}}^T\ ^T$ and the input vector as $\underline{u} =$, eqn. (3.1) can be rewritten as,

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \underline{u}(t), t) \qquad (3.15)$$

where

$\underline{x}(t)$ R^{2n}, $\underline{u}(t)$ R^n, t R^1, $\underline{f} : R^{2n} \times R^n \times R^1$ R^{2n}

and continuously differentiable, and n is the number of degree of freedom of the manipulator.

With this formulation, the objective is to find a feedback control law $\underline{u}(t) = \underline{g}(\underline{x}(t))$ such that the closed loop control system $\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \underline{g}(\underline{x}(t)), t)$ is asymptotically stable and tracks a desired trajectory.

Given a nominal trajectory, nominal torque $u_n(t)$ can be computed using nominal state $x_n(t)$. Then, $u_n(t)$ and $x_n(t)$ satisfy eqn. (3.15), or

$$\dot{\underline{x}}_n(t) = \underline{f}(\underline{x}_n(t), \underline{u}_n(t), t) \qquad (3.16)$$

Linearizing the model of eqn. (3.15) about the nominal trajectory using Taylor's series expansion, and discretizing the continuous time model, perturbation equation is obtained in the following form.

$$\underline{x}(k+1) = A(k)\ \underline{x}(k) + B(k)\ \underline{u}(k) \qquad (3.17)$$

where $\underline{x} = \underline{x} - \underline{x}_n$, $\underline{u} = \underline{u} - \underline{u}_n$ and A(k) and B(k) are (2n x 2n) and (2n x n) matrices respectively. With this model, $6n^2$ parameters are identified.

3.5.2 CONTROLLER DESIGN:

The following performance criterion is selected.

$$J(k) = \frac{1}{2}\ \underline{x}^T(k+1)\ Q\ \underline{x}(k+1) + \underline{u}^T(k)\ R\ \underline{u}(k) \qquad (3.18)$$

The optimal control input $\underline{u}(k)$ which minimizes J(k) while satisfying (3.17) is given by

$$\underline{u}(k) = -K(k)\ \underline{x}(k) \qquad (3.19)$$

where

$$K(k) = \left[R + B^T(k)\ Q\ B(k)\right]^{-1} B^T(k)\ Q\ A(k) \qquad (3.20)$$

The block diagram of the controller is shown in Figure 3.1.

Similar algorithm has been developed by Y.K. Choi et.al 1985 where the joint interactions are taken into account. The scheme is devised based on the Lyapunov direct method which generates a variational control that regulates the perturbation in the vicinity of a desired trajectory.

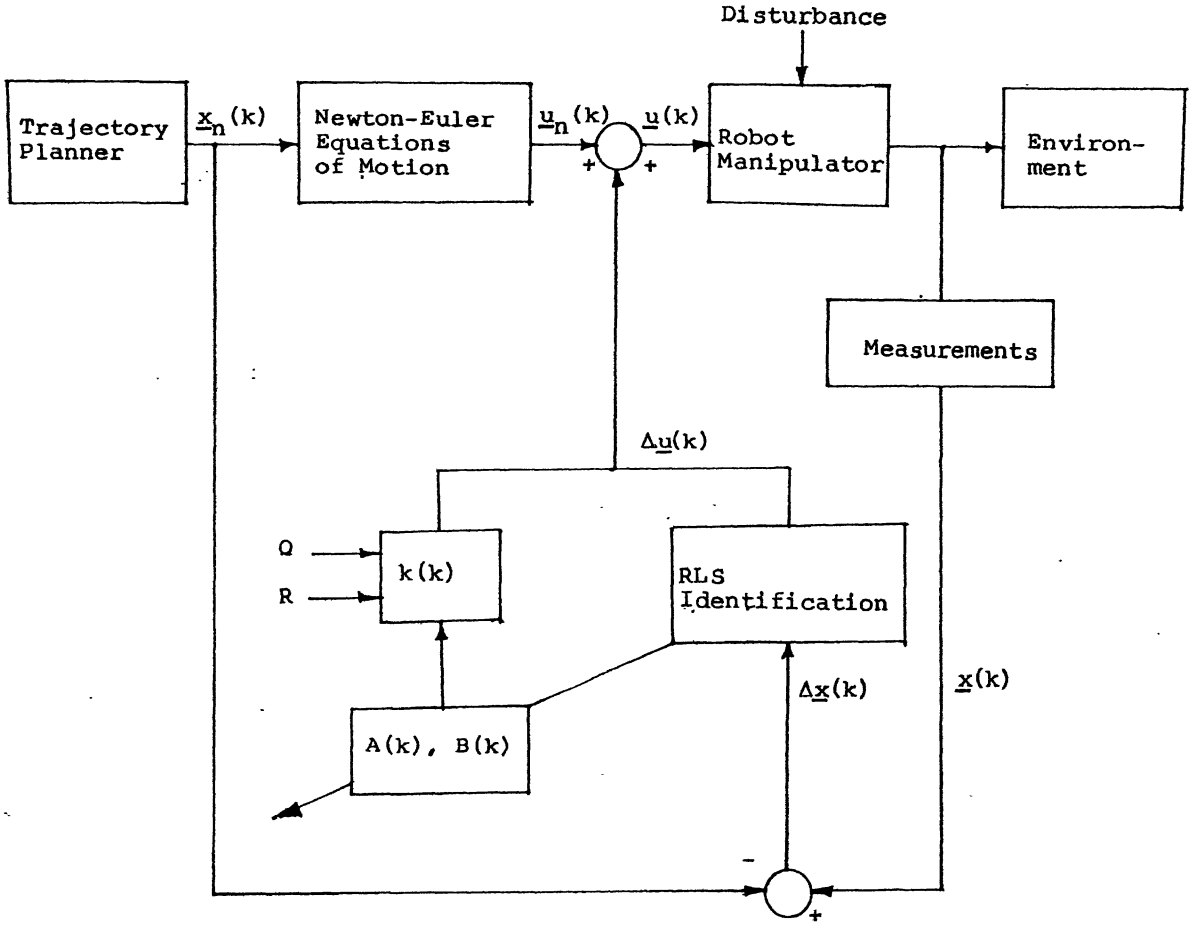


Fig. 3.1: Adaptive Perturbation Control.

Both the algorithms have been tested on PUMA manipulator through digital computer simulations.

3.6 THE NEW ALGORITHMS:

The work presented in the next three sections is carried out essentially in search of a new algorithm that would prove an improvement over the algorithms based on Multi-Input Multi-Output (MIMO) models. Usually, the algorithms based on MIMO models are inefficient in the sense that they involve too many computations per sample and it is difficult to partition these algorithms for multiprocessing.

Following features are desired of the new algorithm:

(1) It should be based on a simplified model. Multiprocessing, if required, should be easy to implement. Arrangement of one processor per joint would be ideal.

(2) The controller should be able to take the robot arm through the required trajectory at speeds comparable to those achieved by the algorithms which use MIMO model.

(3) The design should be such that the control strategy yields a reasonable performance over wide range of manipulator motions and payloads.

(4) It should work on PUMA manipulators.

The algorithm may sacrifice some performance while achieving these requirements. Obviously, there is some trade-off between modeling and tracking accuracy. For simplified models, it is expected that the tracking accuracy will degrade.

3.7 ADAPTIVE PID CONTROLLER (PIDC) FOR ROBOT MANIPULATOR:

Consider a manipulator with degree of freedom equal to n.

Regarding PID controller, it is stated Asada, Slotine, 1985 that the torque for each joint can be computed by the relation

$$j = K_{P_j} e_j + K_{D_j} \dot{e}_j + K_{I_j} \int_0^t e_j \, dt \qquad (3.21)$$

$$j = 1,\ldots,n$$

where $e_j = q_{d_j} - q_j$ is the position error, and coefficients K_{P_j}, K_{D_j} and K_{I_j} are positive and "sufficiently large".

The above equation is expected to control the manipulator in point-to-point positional control.

Effects of gravity are considered separately. At each step, identification and control computation is done without taking gravity into account. While giving computed control to the manipulator, nominal value of the gravity at that position is added to it.

3.7.1 THE ALGORITHM:

In this section, the proposed adaptive PID control algorithm is presented. This algorithm is a straightforward application of the algorithm discussed in Section 2.6. Each joint is considered as a separate entity.

Let the model be described as,

$$A_j(z^{-1})\, y_j(k) = z^{-1} B_j(z^{-1})\, u_j(k) + A_j(z^{-1})\, n_j(k) \qquad (3.22)$$

$$j = 1,\ldots,n$$

$\eta \rightarrow$ eta

where

$u_j(k)$ = T_j at time instant k

$y_j(k)$ = q_j at time instant k.

The control is computed from

$$R_j(z^{-1})\ u(k) = \alpha_j e_j(k) - \beta_j(z^{-1})\ y_j(k) \qquad (3.23)$$

$$j = 1,\ldots,n$$

where

$y_{rj}(k)$ = q_{dj}, the desired position of joint j at time k)

and $e_j(k)$ = $y_{rj}(k) - y_j(k)$.

The closed loop poles are placed at $A_{mj}(z^{-1})$, where

$$A_{mj}(z^{-1}) = 1 + A_{m1j}\ z^{-1} + A_{m2j}\ z^{-2} + A_{m3j}\ z^{-3} + A_{m4j}\ z^{-4} \qquad (3.24)$$

The control method is applied to each joint.

3.8 GENERALIZED SELF-TUNING CONTROLLER (GSTC) WITH POLE PLACEMENT:

The algorithm proposed here uses combination of LQG and pole placement methods.

Consider an n-jointed robot arm. For each joint j, j = 1,...,n, let the model be given by

$$A_j(z^{-1})\ y_j(k) = z^{-d}\ B_j(z^{-1})\ u_j(k) + h_j + e_j(k) \qquad (3.25)$$

where

y_j and u_j are output and input of joint j

e_j is the disturbance term

h_j is the forcing term which includes the effects of the gravitational forces

$$A_j(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a}$$

$$B_j(z^{-1}) = b_o + b_1 z^{-1} + \ldots + b_{n_a-1} z^{-(n_a-1)}$$

n_a is the order of the model.

The backward shift operator and the subscript j are omitted here on for brevity and the following discussion applies to every joint.

The difference equation model is then represented as

$$Ay(k) = z^{-d} Bu(k) + h + e(k) \qquad (3.26)$$

Let the performance criterion be given by

$$I = E\{P(y(k+d) - y_r(k+d))^2 + Q' u(k)^2\} \qquad (3.27)$$

where

$$P = p_o + p_1 z^{-1} + \ldots + p_{n_p} z^{-n_p}$$

$$Q' = q'_o + q'_1 z^{-1} + \ldots + q'_{n_q} z^{-n_q}$$

It has been shown [Clarke, Gawthrop, 1975] that minimizing I and $J = E\{\phi^2(k+d)\}$ will give same control law, where

$$\phi(k+d) = P[y(k+d) - y_r(k+d)] + Q u(k) \qquad (3.28)$$

and $Q = (q'_o/b_o)Q'$

Combining (3.26) and (3.28)

$$\phi(k+d) = \frac{P}{A} B u(k) + h + \frac{P}{A} e(k+d) + Q u(k) - P y_r(k+d)$$

(3.29)

Let

$$\frac{1}{A} = F + z^{-d}\frac{G}{A} \tag{3.30}$$

where F and G are polynomials in z^{-1} with the orders d-1 and n_a-1 respectively.

To minimize the performance criterion J, the control law u(k) should be chosen M.H. Liu, 1985 such that

$$H\,u(k) + G\,y(k) - P\,y_r(k+d) + r = 0 \tag{3.31}$$

where

$$H = BF + Q \tag{3.32}$$

$$r = F(z^{-1})\,h \tag{3.33}$$

A method of an line determination of the costing polynomials P and Q is now presented according to closed-loop pole assignment.

Applying the control law u(k) determined by eqn. (3.31) to system in eqn. (3.26) yields the closed-loop equation

$$y(k) = \frac{BP}{BP + AQ}\,y_r(k) + \frac{Q}{BP + AQ}\,h + \frac{H}{BP + AQ}\,e(k) \tag{3.34}$$

Choose P and Q such that

$$BP + AQ = A_m \tag{3.35}$$

where A_m is a prespecified polynomial in z^{-1} and the orders of the polynomials P, Q and A_m are

$$n_p = n_a - 1, \quad n_q = n_a - 2, \quad n_m = 2n-1$$

respectively.

3.8.1 THE ALGORITHM:

The control algorithm can be summarized as follows.

Data : B_a, Polynomial A_m, Initial values of polynomials P and h.

Step 1: Estimate the polynomials A, B and h using RLS.

Step 2: Calculate P, h from (3.35).

Step 3: Calculate H, G, F and r from (3.30), (3.32) and (3.33).

Step 4: Get control law u(k) from (3.31).

Repeat steps 1, 2, 3 and 4 for each sampling interval.

3.9 DECENTRALIZED ADAPTIVE PERTURBATION CONTROL (DAPC):

The basic idea of the perturbation control is presented in Section 3.5.

The block diagram of decentralized control using perturbation approach is shown in Figure 3.2.

Each joint of the manipulator is considered as a separate entity. A linear regression single input single output (SISO) model is assumed for each joint. The joint position is considered as output. Parameter estimation is done based on this model using RLS method.

Experiments have been done with the following two control strategies mentioned in Chapter 2:

(1) M.V. control

(2) Adaptive PID control.

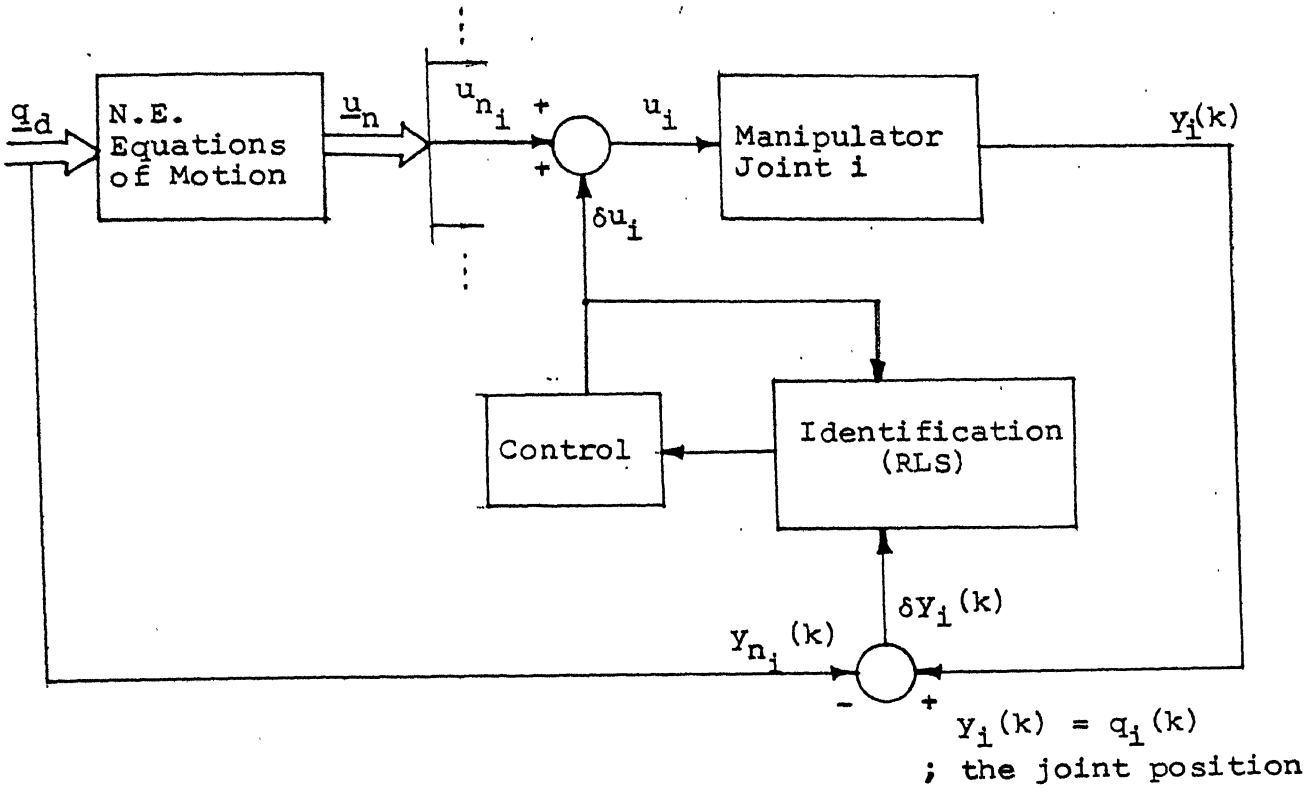


Fig. 3.2: Decentralized Adaptive Perturbation Control.

3.10 DIGITAL SIMULATION:

The algorithms PIDC, GSTC and DAPC have been tested on PUMA-600 through digital simulation.

In all the experiments, same trajectory has been used. The trajectory is generated using Paul's algorithm [R.P.C. Paul, 1981].

It is desired to move the robot arm from initial position $\underline{q}_i$ to final position $\underline{q}_f$, where

$$\underline{q}_i = [0^\circ \quad 45^\circ \quad 45^\circ \quad 45^\circ \quad -45^\circ \quad -180^\circ]^T$$

$$\underline{q}_f = [90^\circ \quad -45^\circ \quad 135^\circ \quad 105^\circ \quad 15^\circ \quad -120^\circ]^T$$

The maximum velocity of each joint is specified by a vector $\underline{V}_{max}$.

$$\underline{V}_{max} = 80^{\circ}/\text{sec} \quad 35^{\circ}/\text{sec} \quad 120^{\circ}/\text{sec} \quad 70^{\circ}/\text{sec} \quad 60^{\circ}/\text{sec} \quad 90^{\circ}/\text{sec}^{T}$$

The time required to acquire maximum velocity starting from zero velocity is set to 0.1 sec. for all joints.

The sampling interval is selected to be 0.016 sec.

It is observed that these algorithms do not give satisfactory performance. In fact, the parameters of the robot arm do not converge and this results in failure of the control strategy.

3.11 CRITIQUE:

It is possible that the failure in PIDC, GSTC and DAPC algorithms were due to the following reasons:

(1) Non-minimum phase behavior of a manipulator joint.

(2) Imperfect modeling of the manipulator.

(3) Inaccurate initialization.

They are elaborated below.

(1) Non-minimum phase behavior:

In PID self-tuner, control parameters are calculated directly by using RLS method. Hence, nothing can be said about the system zeros. It has been seen in Chapter 2 that M.V. control and PID self-tuner fail to control a system which shows non-minimum phase behavior. Hence, if these strategies do not give satisfactory performance, it is justified to express a possibility of system being non-minimum phase. In general, in the linearized model it is difficult to guarantee minimum phase behavior. Furthermore, in robotic systems, minimum phase

requirement is satisfied in general only if the manipulator path is well defined such that the dynamics, as seen at each joint, are essentially inertial quantities within well defined ranges Leininger, Wang, 1982 . In the experiments, no care has been taken while planning the path so as to satisfy this requirement as this would be a stringent requirement on the path planner.

In simulation of DAPC, it was indeed observed that the system shows non-minimum phase behavior.

(2) Imperfect modeling of the manipulator:

Of the six controllers discussed, PPST, ALC, PIDC, GSTC, DAPC do not include terms for the interactions between the joints. Even though the experiments with PIDC, GSTC and DAPC failed on the PUMA manipulator, PPST and ALC have been reported as successful on JPL-Stanford manipulator in Leininger, Wang, 1982 and Koivo, Guo, 1983 respectively. This, however, raises the possibility that there are heavy joint interactions in PUMA manipulators, particularly as the only controller known to have worked on PUMA is DAPC which includes terms for interaction forces.

Returning to the eqn. (3.1), a typical D-matrix of PUMA-600 is presented here. This is obtained for

$$\underline{q} = \begin{bmatrix} 0.14 & 44.9 & 45.2 & 44.9 & -45.0 & -176.8 \end{bmatrix}^T$$

$$\dot{\underline{q}} = \begin{bmatrix} 6.35 & 6.53 & 8.59 & 2.0 & 2.8 & -1.77 \end{bmatrix}^T$$

$$\ddot{\underline{q}} = \begin{bmatrix} 258.0 & -256.0 & -73.9 & 528.4 & 418.2 & 576.9 \end{bmatrix}^T$$

The units of elements in $\underline{q}$, $\dot{\underline{q}}$ and $\ddot{\underline{q}}$ are degrees, degrees/sec and degrees/sec^2 respectively.

374.79	-32.24	1.11	1.49	0.232	0.015
-32.24	27.62	3.676	0.488	0.231	-0.018
1.11	3.676	4.05	0.52	0.23	-0.02
1.49	0.488	0.52	0.494	0	0.0264
0.232	0.231	0.23	0	0.3275	0
0.015	-0.018	-0.02	0.0264	0	0.0375

If the diagonal terms were predominant, it could have been argued that even if one neglects the off-diagonal terms, it would not make any difference to the individual joints. However, it is observed that there are high joint interactions in PUMA-600. For example, effect of $\ddot{\theta}_1$ on $_2$ is more than that of $\ddot{\theta}_2$. In other words, if effect of $\ddot{\theta}_1$ on $_2$ is neglected, we have a model in which noise is dominant over the signal!

(3) Inaccurate initialization:

All the decentralized models are extremely sensitive to the initial guess and this causes great difficulties in the starting of the algorithms. Consider a centralized control method, CAPC, in which an n-jointed robot arm is modeled with $6n^2$ parameters. But in PIDC, a decentralized approach, only 4n parameters are used to model the robot arm. For a six jointed robot arm, these numbers are 216 and 24 respectively. With such a small number of parameters available to model the arm, it is extremely difficult to make an intelligent guess of the initial values in case of decentralized approach. Moreover, in MIMO model used for centralized control, parameters can be initialized using the robot dynamic model itself based on the starting

position of the arm. There is no way to find relationship between parameters of decentralized control and the elements of D, H, G matrices in the dynamic model of the robot arm. The dynamic effects are to be estimated without using any knowledge about system dynamics Vukobratovic et.al., 1985 . Thus, it is impossible to make a calculated guess of initial parameters. Even after initializing to some arbitrary values, the parameters are estimated using 'black box concept' without any knowledge of the system. Thus, if initial guess is far from actual parameters, the parameters may not converge in subsequent recursion.

Thus, to achieve control using a decentralized model, many trial and error iterations have to be done. It calls for a good deal of experience and insight to choose initial values which will subsequently lead to currect identification.

3.12 DISCUSSION:

All decentralized controllers have a clear drawback in that they neglect the joint interactions. Apart from that, there are many vital points which deserve some discussion.

The Minimum Variance procedure is essentially a pole placing algorithm where the controller places poles/zeros to cancel system zeros/poles - i.e. the closed loop poles are placed at the origin. The resulting controller is, however, erratic and often has excessively large amplitudes. This strategy cannot be applied if identified system zeros are outside the unit circle.

Several problems have been reported with LQG control as applied to robotic manipulators. The control law can result in

a closed-loop unstable operation if actual robot parameters are far from the initial guess and/or are rapidly time-varying compared to adaption speed. Furthermore, proper selection of Q and R matrices compound the design problem Vukobratovic et. al., 1985 . It has also been reported Koivo, Guo, 1983 that estimated parameters change erratically during the first part of the trajectory producing abrupt variations in control signals and erratic movements of the robot arm. The results are obtained after much trial and error iterations concerning initial guess and performance index weighting factors. Moreover, the results are obtained through simulation. In real situation, it remains to be seen how much erratic motions the actuators can take. Also, the results obtained with trial and error are applicable for repetitive task in a well defined trajectory. These results do not generalize to a wide range of manipulator motions and payloads.

The conclusions drawn for LQG approach practically hold for pole-zero placement control as well. The difference is that closed-loop poles are imposed instead of weighting matrices. Moreover, this procedure is described for single input single output models only.

With reference to the algorithms discussed, PPST will fail if controller polynomial G has zeros outside unit circle. Similarly, if matrix R has unstable eigen values, ALC will fail.

The PID self tuning regulator has also been attempted by Vukobratovic et.al., 1985 . They have reported that the behavior of the manipulator when starting the learning process depends mostly on the accuracy in initialization of the predictio

model parameters. This accuracy determines the degree of erratic motions at start up. The convergence of parameter estimates and controller gains may not be achieved during the finite time over which motion takes place. Also, when the parameters change abruptly, the staoility of overall system may fail.

Adaptive Perturbation Control [Lee, Chung, 1984] is a centralized scheme based on Multi-Input Multi-Output (MIMO) model. In this scheme, nominal torque values are computed on-line. It has been reported by the authors that the performance of the adaptive controller is quite sensitive to values of Q, R in eqn. (3.18). In general it is not easy to determine the proper values to acnieve better performance. It is also difficult to figure out the correlations of these values with the overall performance of the controller especially in MIMO system because of the complexity of the dynamic equations of the PUMA robot arm. There is one more problem with this approach which is still unresolved. First, suppose that the parameters of a robotic system are exactly known. Then it is obvious that the identification of the linearized model is not necessary. On the other hand, if some parameters are unknown (for example, work piece mass and inertias), then the nominal control cannot be calculated exactly. Thus it is necessary to introduce an adaptive algorithm at nominal level as well. The adaptive feedforward compensation is not discussed in [Lee, Chung, 1984] and [Choi et.al., 1985].

CHAPTER 4

ADAPTIVE CONTROL ANALYSIS

In Chapter 3, it is seen that it is virtually impossible to design a controller based on SISO model with features mentioned in Section 3.6. Thus, one has to resort to a model which takes joint interactions into account.

In this chapter, first Computed Torque Adaptive Control Patnaik, 1987 , which is essentially a centralized control strategy is presented. This model assumes availability of joint position and velocity measurements. Next, a model is derived which uses only position measurements. Lastly, the above two models are analysed. In the analysis, the effects of neglecting some system dynamic parameters, as well as, the effects of measurement noise on the performance of the adaptive controllers are discussed.

4.1 COMPUTED TORQUE ADAPTIVE CONTROL Patnaik, 1987 :

The Computed Torque Adaptive Control algorithm is basically an adaptive version of computed torque control which is also called the inverse problem.

4.1.1 THE COMPUTED TORQUE CONTROL Asada, Slotine, 1985 :

The dynamic equations of a robot manipulator are given by,

$$D(\underline{q})\,\ddot{\underline{q}} + \underline{H}(\underline{q}, \dot{\underline{q}}) + \underline{G}(\underline{q}) = \qquad (4.1)$$

where,

$$\underline{H}(\underline{q}, \dot{\underline{q}}) = H_1(q) \begin{bmatrix} \dot{q}_1^2 \\ \dot{q}_1\dot{q}_2 \\ \vdots \\ \dot{q}_1\dot{q}_n \\ \dot{q}_2^2 \\ \vdots \\ \dot{q}_2\dot{q}_n \\ \dot{q}_3^2 \\ \vdots \\ \dot{q}_n^2 \end{bmatrix} \tag{4.2}$$

$H_1(\underline{q})$ is an $n \times \frac{n(n+1)}{2}$ matrix.

Suppose the desired torque is computed from

$$\underline{T} = D_c(\underline{q})\ \ddot{\underline{q}}_d + k_1(\dot{\underline{q}}_d - \dot{\underline{q}}) + k_2(\underline{q}_d - \underline{q}) + \underline{H}_c(\underline{q}, \dot{\underline{q}}) + \underline{G}_c(\underline{q}) \tag{4.3}$$

where k_1 and k_2 are diagonal gain matrices, $\underline{q}_d$ is a desired position vector and D_c, $\underline{H}_c$ and $\underline{G}_c$ are calculated or measured counterparts of D, $\underline{H}$ and $\underline{G}$, respectively.

Assuming that

$$D_c(\underline{q}) = D(\underline{q}) \tag{4.4}$$

$$\underline{H}_c(\underline{q}, \dot{\underline{q}}) = \underline{H}(\underline{q}, \dot{\underline{q}}) \tag{4.5}$$

$$\underline{G}_c(\underline{q}) = \underline{G}(\underline{q}) \tag{4.6}$$

Eqns. (4.1) and (4.3) can be combined to obtain

$$D(\underline{q})\ \ddot{\underline{q}}_d - \ddot{\underline{q}} + k_1(\dot{\underline{q}}_d - \dot{\underline{q}}) + k_2(\underline{q}_d - \underline{q}) = \underline{0} \qquad (4.7)$$

Defining the position error vector

$$\underline{e}_q = \underline{q}_d - \underline{q} \qquad (4.8)$$

Eqn. (4.7) becomes

$$\ddot{\underline{e}}_q + k_1 \dot{\underline{e}}_q + k_2 \underline{e}_q = \underline{0} \qquad (4.9)$$

If k_1 and k_2 are so chosen that the characteristic roots of eqn. (4.9) are on negative half of the s-plane, then the error $\underline{e}_q$ will come down to $\underline{0}$ asymptotically. The convergence relies on validity of eqns. (4.4), (4.5) and (4.6).

4.1.2 ADAPTIVE CONTROL:

Due to parameter uncertainties, various frictional forces, modeling errors, etc., it is very difficult to satisfy eqns. (4.4)-(4.6). This difficulty is solved using adaptive control in which the dynamic parameters are identified on line. Let

$$A = D^{-1} \qquad (4.10)$$

$$\underline{B} = D^{-1}\,\underline{G} \qquad (4.11)$$

$$C = D^{-1}\,H_1 \qquad (4.12)$$

Discretizing eqn. (4.1) using backward rectangular rule, the following equation is obtained.

$$\underline{Y}(k) = \underline{Y}(k-1) + T\,A\,\underline{U}(k-1) - T\,\underline{B} - T\,C\,\underline{Y}_c(k-1) \qquad (4.13)$$

where

$\underline{Y}$ is the joint velocity (output) vector,

$\underline{U}$ is the control (input) vector,

T is the sampling period,

and

$$\underline{Y}_c(k-1) = \begin{bmatrix} y_1^2(k-1) \\ y_1(k-1)\ y_2(k-1) \\ \vdots \\ y_1(k-1)\ y_n(k-1) \\ y_2^2(k-1) \\ \vdots \\ y_2(k-1)\ y_n(k-1) \\ y_3^2(k-1) \\ \vdots \\ y_n^2(k-1) \end{bmatrix} \qquad (4.14)$$

The control is computed from

$$\underline{U}(k) = A^{-1} \cdot \ddot{\underline{q}}_d(k) + k_1\, \dot{\underline{e}}_q(k) + k_2\, \underline{e}_q(k) + \underline{B} + C\, \underline{Y}_c(k) \qquad (4.15)$$

where A, $\underline{B}$ and C are obtained on-line using system identification.

System Identification:

To find the parameters of eqn. (4.13) a multivariable version of Least Squares method has been used. Defining,

$$A' = TA, \quad \underline{B}' = T\underline{B}, \quad C' = TC \qquad (4.16)$$

Eqn. (4.13) is rewritten as,

$$\underline{Y}(k) = \underline{Y}(k-1) + A'\,\underline{U}(k-1) - \underline{B}' - C'\,\underline{Y}_c(k-1) + \underline{\varepsilon}(k) \tag{4.17}$$

where $\underline{\varepsilon}(k)$ is $(n \times 1)$ vector representing the error in the modeling. Let

$$\theta^T = I \vdots A' \vdots B' \vdots C' \tag{4.18}$$

$$\underline{\psi}^T(k) = \underline{Y}^T(k-1) \vdots \underline{U}^T(k-1) \vdots -1 \vdots -\underline{Y}_c^T(k-1) \tag{4.19}$$

Then, eqn. (4.17) can be written as

$$\underline{Y}(k) = \theta^T\,\underline{\psi}(k) + \underline{\varepsilon}(k) \tag{4.20}$$

The prediction equation is then given by

$$\underline{Y}(k/\theta) = \theta^T\,\underline{\psi}(k) \tag{4.21}$$

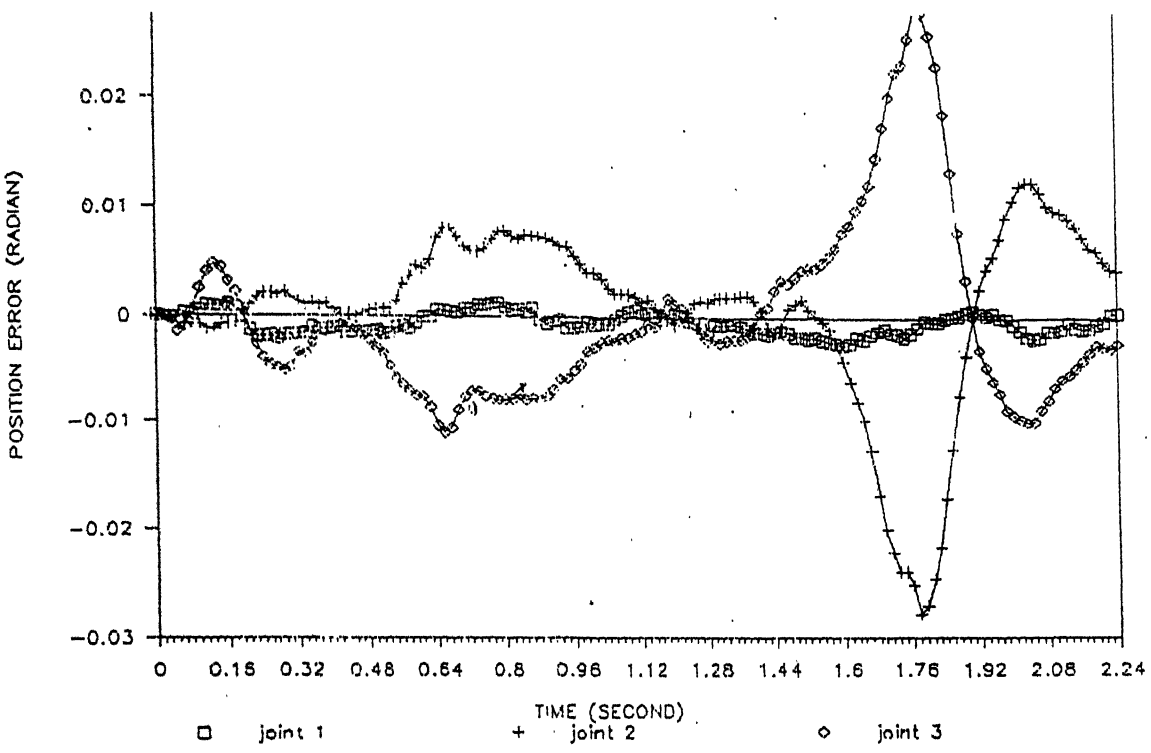
Using eqns. (4.20) and (4.21), the parameter vector is update in a straightforward manner.

4.1.3 DIGITAL COMPUTER SIMULATION:

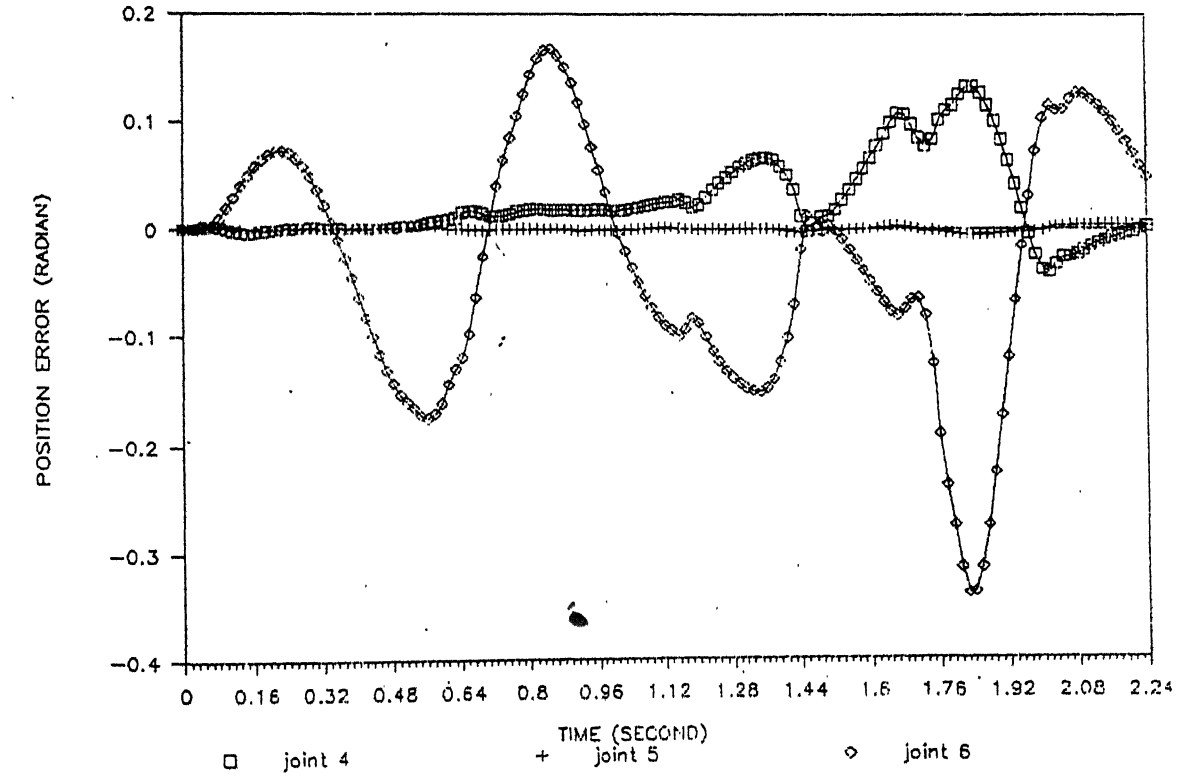
The Computed Torque Adaptive Control algorithm has been tested for the first three joints of PUMA manipulator Patnaik, 1987 .

To test the algorithm on a six-jointed manipulator, simulation of PUMA-600 robot arm has been done on DEC-1090. Trajectory for the robot arm is planned using Paul's algorithm Paul, 1981 . Details of the trajectory are given in Section 3.10.

The plots of errors between the desired and the actual joint positions as function of time are shown in Figure 4.1. It can be seen that the algorithm gives satisfactory performance for a six-joint robot arm.



(a)



(b)

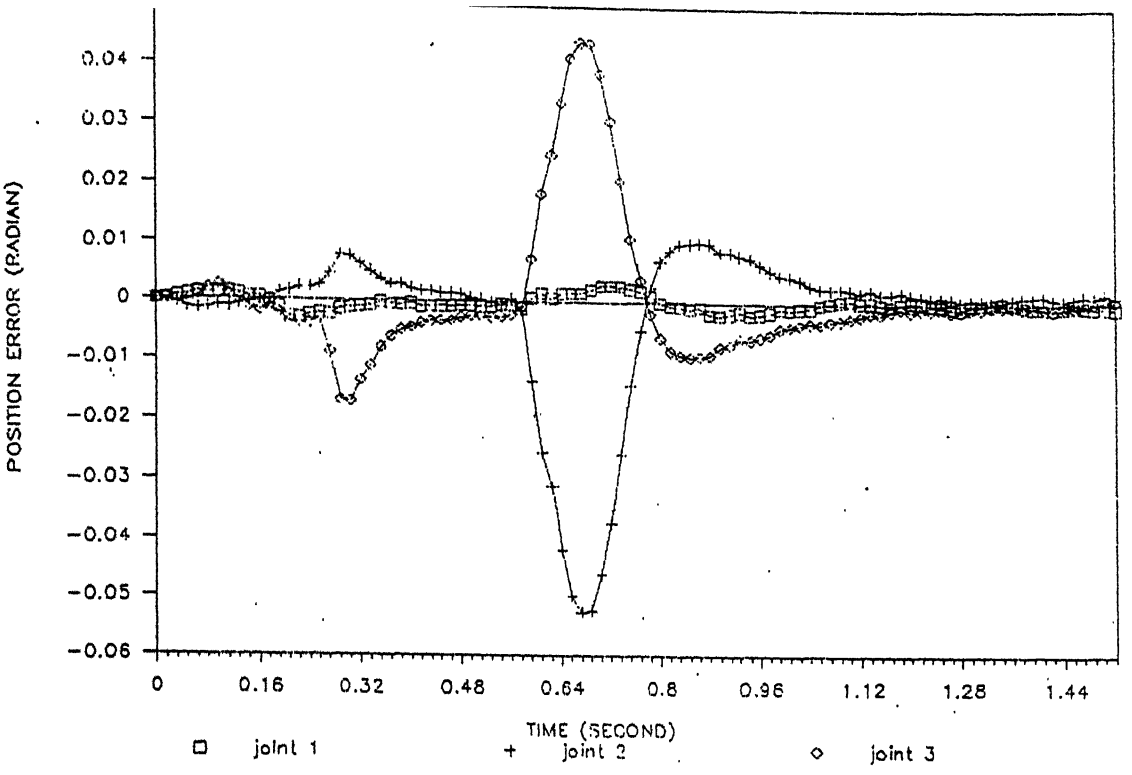
Fig. 4.1: Tracking Error (maximum linear speed 1.5 m/s)
(a) First three joints
(b) Last three joints

It must be noted that the Computed Torque Adaptive Control algorithm is superior to the Centralized Adaptive Perturbation Control in many respects. In Section 3.12, it has been seen that the performance of Centralized Adaptive Perturbation Control is quite sensitive to values of Q and R matrices in eqns. (3.17) and (3.20). In Computed Torque Adaptive Control, the parameters can be initialized in a straightforward manner once the initial position of the robot is known. Moreover, it does not have to compute nominal torque values, thus eliminating the need of adaptive feedforward compensation.

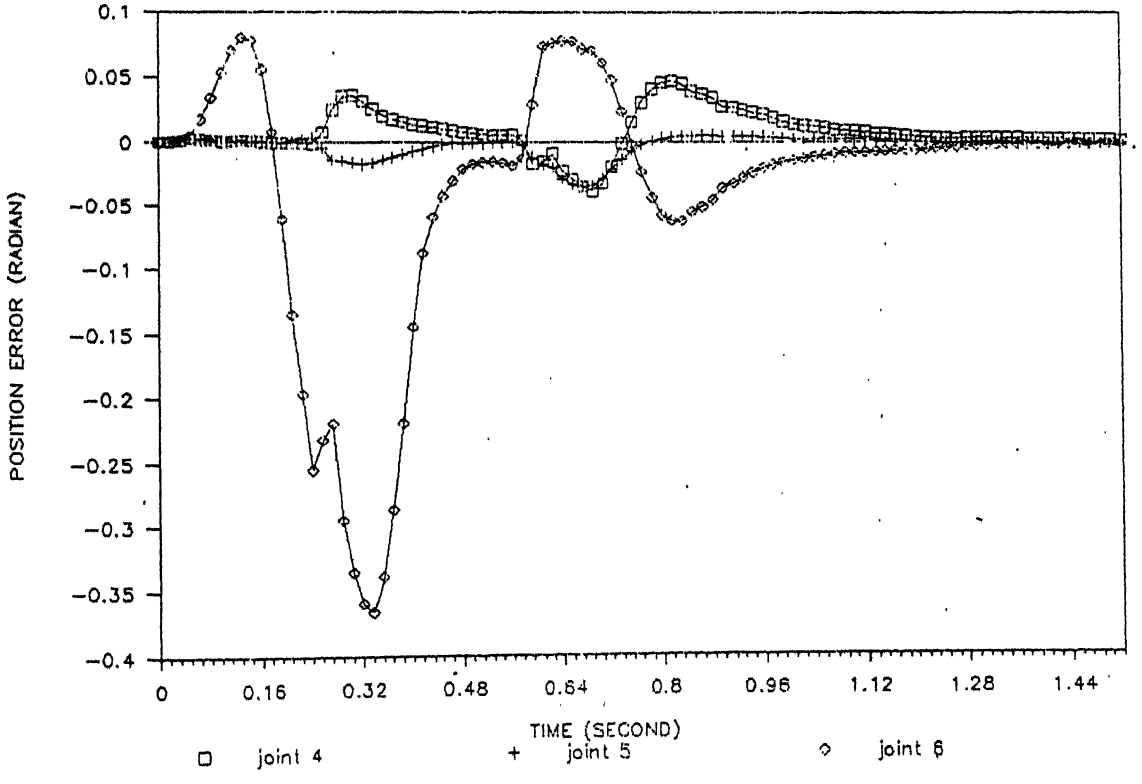
It has been observed that the controller is able to take the robot arm through the trajectory in which the maximum linear speed is 4.25 m/s. This speed is higher than that reported elsewhere. However, this speed is tested on computer simulation and it remains to be seen how much speed the actuators can tolerate. But as far as algorithm is concerned, it is robust enough to achieve high speed tracking. [illegible] motion at a high speed.

4.1.4 SEPARATION OF ARM AND WRIST:

The main difficulty in implementing this algorithm is that it involves a large amount of computation. A natural way to overcome this difficulty is to neglect interactions between the joints to some extent. First step in this direction would be to decompose the robot model into two groups. The 'arm' (first three joints) and the 'wrist' (last three joints) may be controlled independent of each other. Such a scheme has been used by Koivo, Guo, 1983 and Liu, Chen, 1986 . However, this decomposition is done on the basis of mechanical structure of the manipulator. It is possible in manipulators like JPL-Stanford



(a)



(b)

Fig. 4.2: Tracking Error (maximum linear speed 4.25 m/s)
(a) First three joints
(b) Last three joints

in which presence of prismatic joint reduces the coupling terms between the variables of the two groups significantly. However, in case of PUMA manipulator, observation of D-matrix (shown in Section 3.10) makes it clear that this kind of splitting is not possible. Hence, a full order model has to be used.

4.2 A MODEL WITH JOINT POSITIONS AS OUTPUT:

It has been seen that the algorithm based on Computed Torque Adaptive Control gives satisfactory performance when joint velocities are chosen as model output.

However, in many practical applications, robotic systems are provided with only position sensors. It is then necessary to build a controller that would compute control input based on measurement of joint positions only. In this section, a model is derived with joint positions as output. The validity of this model is tested.

4.2.1 DERIVATION OF THE MODEL:

Consider an n-jointed robot arm. Then the dynamic equations of the arm are given by eqn. (4.1). Let

$$A = D^{-1}$$

$$\underline{B} = D^{-1}\,\underline{G}$$

$$C = D^{-1}\,H_1$$

Then, eqn. (4.1) becomes

$$A\_ = \ddot{\underline{q}} + \underline{B} + C \qquad \begin{matrix} \dot{q}_1^2 \\ \dot{q}_1\dot{q}_2 \\ \vdots \\ \dot{q}_1\dot{q}_n \\ \dot{q}_2^2 \\ \dot{q}_2\dot{q}_3 \\ \vdots \\ \dot{q}_3^2 \\ \vdots \\ \dot{q}_n^2 \end{matrix} \qquad (4.22)$$

Let $\underline{x}(k) = \underline{q}(k)$ and $\underline{U}(k) = \underline{\tau}(k)$.

Eqn. (4.22) is discretized by using forward Euler rule. The position vector at time k becomes

$$\underline{Y}(k) = \underline{x}(k-1) + T\,\dot{\underline{Y}}(k-1) \qquad (4.23)$$

and the velocity is given by

$$\dot{\underline{Y}}(k) = \dot{\underline{Y}}(k-1) + T\,\ddot{\underline{Y}}(k-1) \qquad (4.24)$$

The eqn. (4.22) is modified as,

$$\underline{Y}(k) - 2\underline{Y}(k-1) + \underline{Y}(k-2) = T^2 A\,\underline{U}(k-2) - T^2 \underline{B} - T^2 C\,\underline{V}(k-2) + \_(k) \qquad (4.25)$$

where T is the sampling period,

$\_(k)$ is the modeling error and

$$\underline{v}(k) = \begin{matrix} \dot{q}_1^2(k) \\ \dot{q}_1(k)\ \dot{q}_2(k) \\ \vdots \\ \dot{q}_1(k)\ \dot{q}_n(k) \\ \dot{q}_2^2(k) \\ \dot{q}_2(k)\ \dot{q}_3(k) \\ \vdots \\ \dot{q}_3^2(k) \\ \vdots \\ \dot{q}_n^2(k) \end{matrix} \qquad (4.26)$$

i.e.

$$\underline{v}(k) = \frac{1}{T^2} \begin{matrix} [y_1(k+1) - y_1(k)]^2 \\ [y_1(k+1) - y_1(k)]\,[y_2(k+1) - y_2(k)] \\ \vdots \\ [y_1(k+1) - y_1(k)]\,[y_n(k+1) - y_n(k)] \\ [y_2(k+1) - y_2(k)]^2 \\ \vdots \\ [y_2(k+1) - y_2(k)]\,[y_n(k+1) - y_n(k)] \\ [y_3(k+1) - y_3(k)]^2 \\ \vdots \\ [y_n(k+1) - y_n(k)]^2 \end{matrix} \qquad (4.27)$$

Let

$$A' = T^2 A \tag{4.28}$$

$$\underline{B}' = -T^2 \underline{B} \tag{4.29}$$

$$C' = -T^2 C \tag{4.30}$$

Then, eqn. (4.25) becomes

$$\underline{Y}(k) = 2\underline{Y}(k-1) - \underline{Y}(k-2) + A' \underline{U}(k-2) + \underline{B}' + C' \underline{V}(k-2) + \_(k) \tag{4.31}$$

Eqn. (4.31) can be expressed as,

$$Y(k) = \theta^T \underline{\Psi}(k) + (k) \tag{4.32}$$

where

$$\theta^T = 2I \vdots I \vdots A' \vdots B' \vdots C' \tag{4.33}$$

$$\Psi^T(k) = \underline{Y}^T(k-1) \vdots -\underline{Y}^T(k-2) \vdots \underline{U}^T(k-2) \vdots 1 \vdots \underline{V}^T(k-2) \tag{4.34}$$

The prediction equation is

$$\hat{y}(k/\theta) = \theta^T \underline{\Psi}(k) \tag{4.35}$$

It is to be noted that $\underline{V}(k)$ can be calculated from the joint position measurements. However, if the joint velocity measurements are available, they can directly be used for calculation of $\underline{V}(k)$.

Open loop system identification has been performed to confirm validity of this model. The RLS method has been used to identify the parameter matrix of three jointed PUMA-560 through simulation. It has been seen that system identification is major

bottle-neck of the adaptive control schemes in the sense that if parameters of the system are identified properly, design of a control strategy is comparatively easier.

The simulation results are presented in Figure 4.3. The plots of some of the actual parameters and identified parameters are shown. The plotted parameters are elements of the matrices D, P and Q and their identified counterparts, where,

$$P = D^{-1} A_1 \tag{4.36}$$

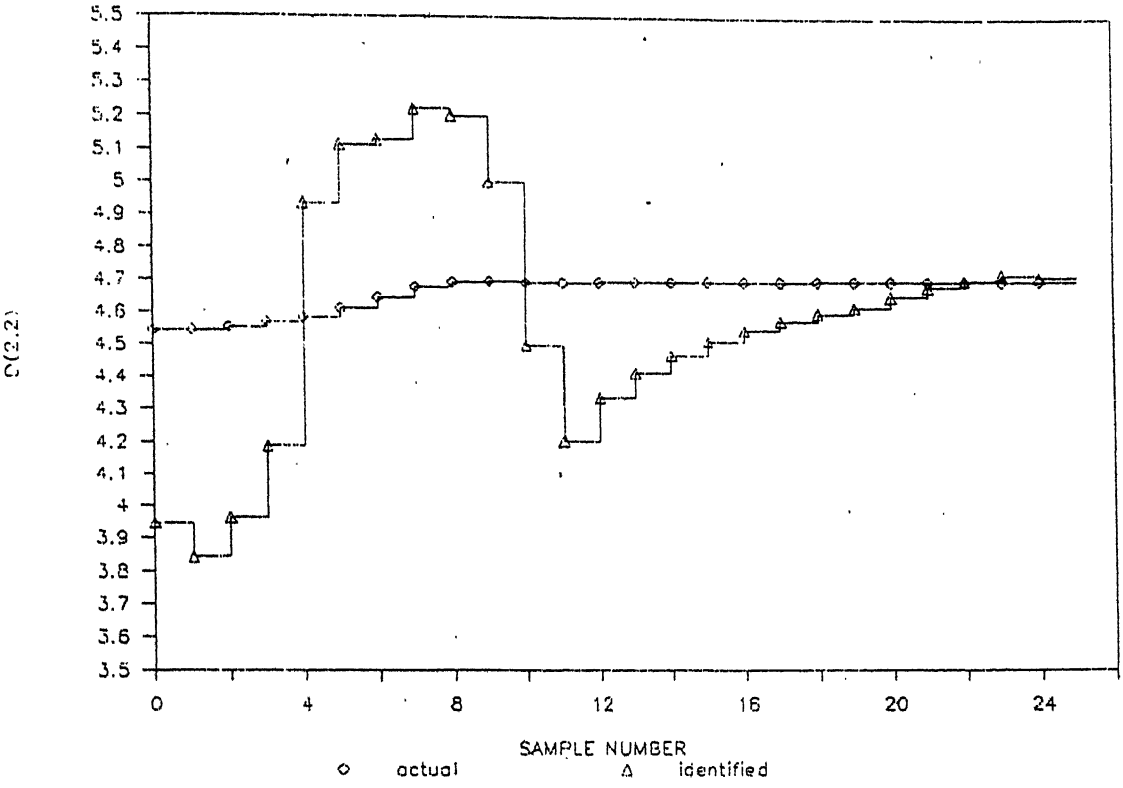
$$Q = D^{-1} G \tag{4.37}$$

It can be seen that the identified parameters track the actual parameters of the model, proving validity of this model. With knowledge of these identified parameters, any of the techniques discussed in Chapter 2 can be successfully applied to achieve control. Obviously, the algorithm developed in Chapter 2 will then have to be reconstructed to work on MIMO system. The control part is not tested.

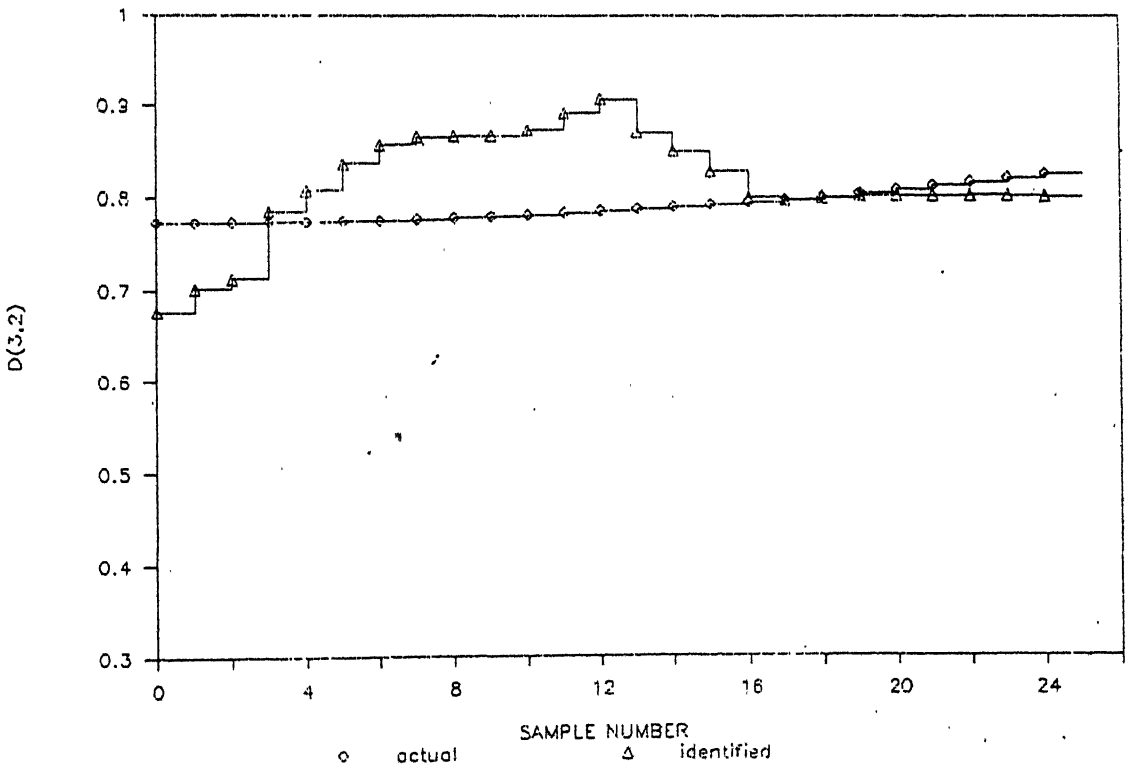
4.3 EFFECT OF MEASUREMENT NOISE:

To test how the model derived in the previous section will work under noisy conditions, measurement noise is added to the output. Under this condition eqn. (4.31) is no longer valid, and modifications are required in the governing equation.

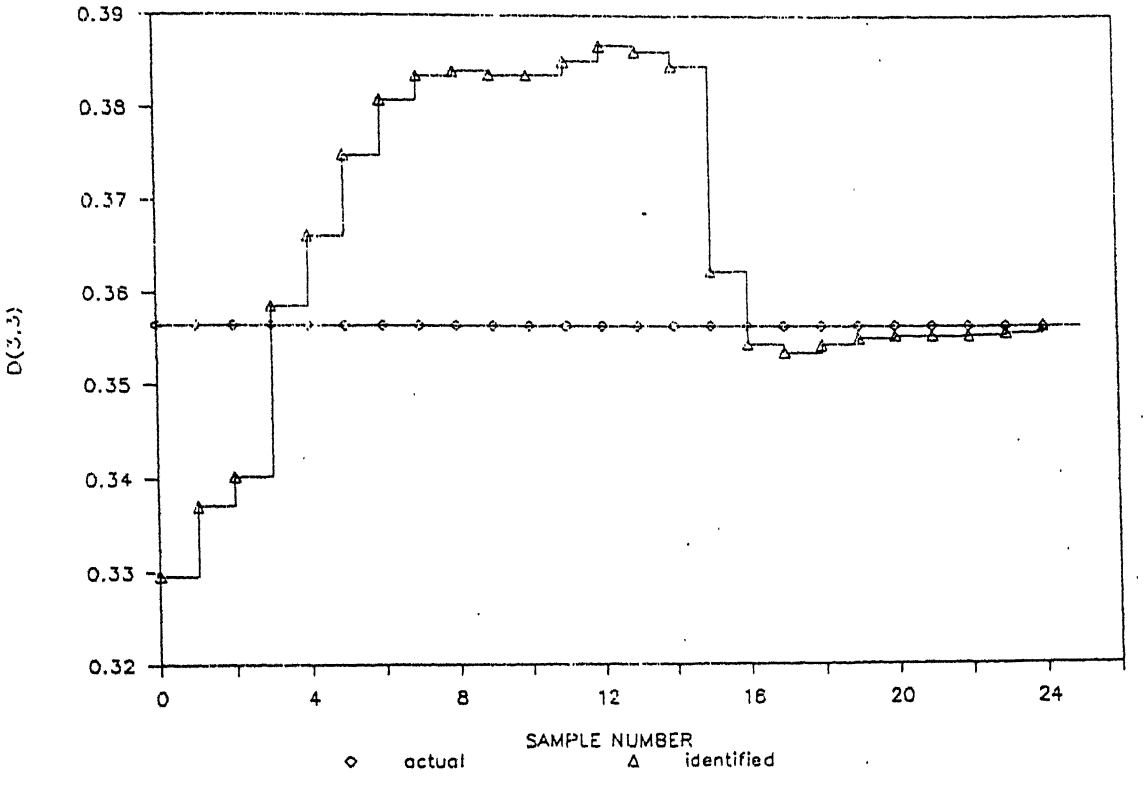
Consider the model derived in Section 4.2. The model (4.31) is reproduced for two jointed robot arm for simplicity.


C(2,2)
5.5
5.4
5.3
5.2
5.1
5
4.9
4.8
4.7
4.6
4.5
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.6
3.5
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
△ identified

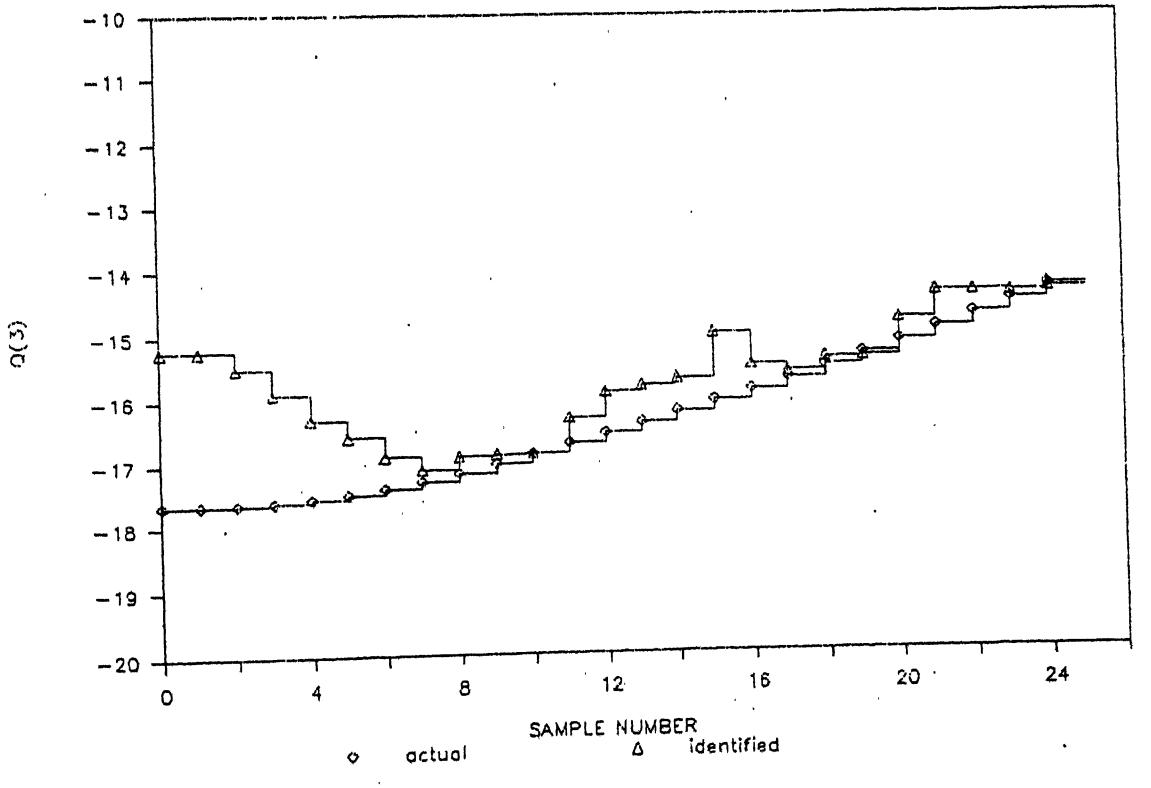
(a)


D(3,2)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
△ identified

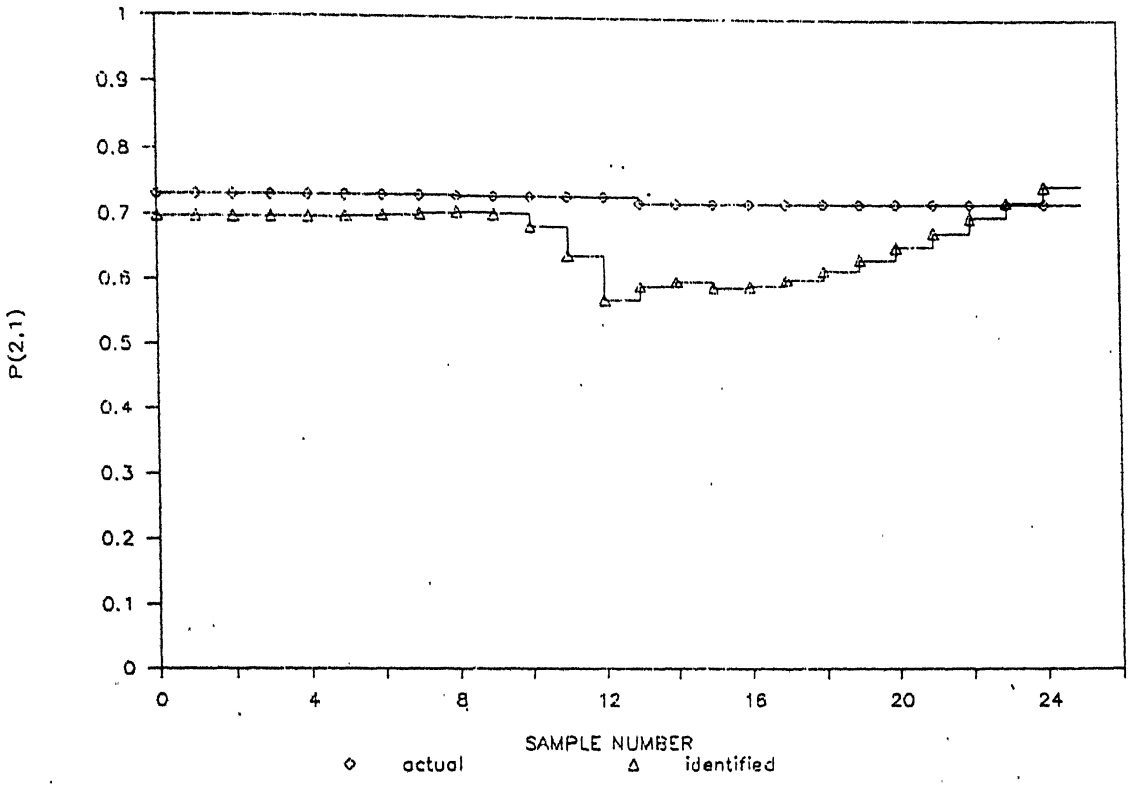
(b)


0.39
0.38
0.37
0.36
0.35
0.34
0.33
0.32
D(3,3)
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

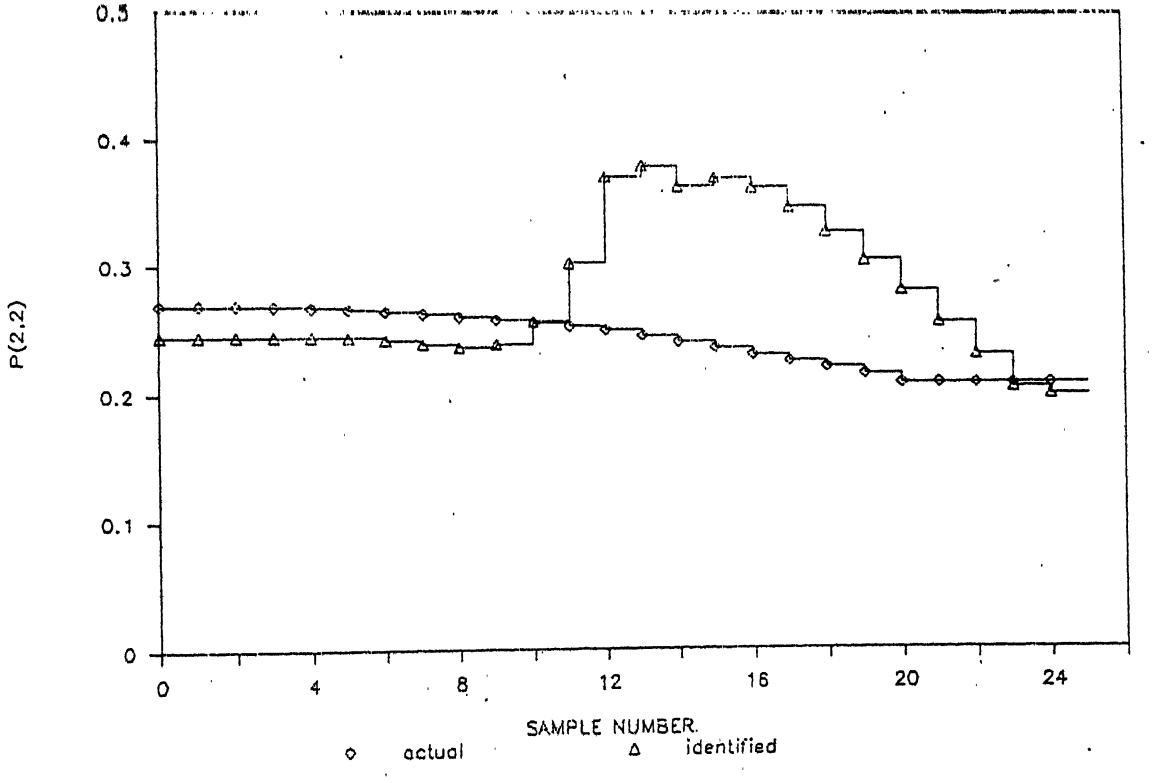
(c)


-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
Q(3)
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

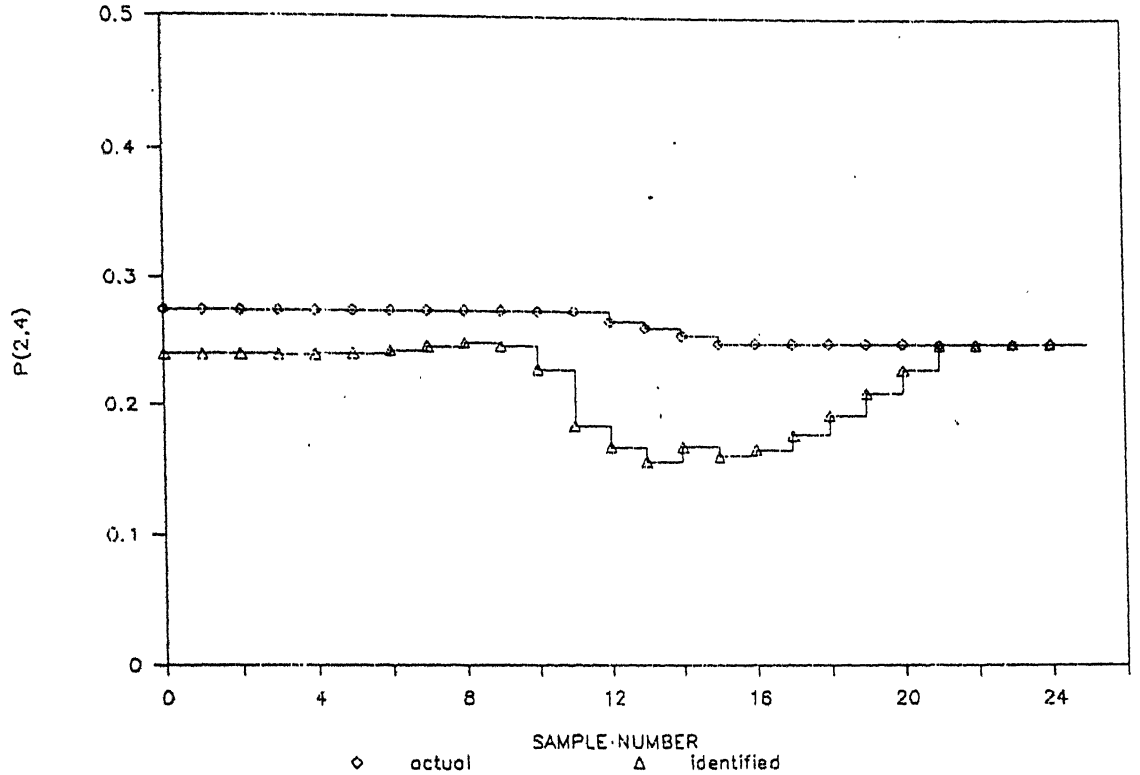
(d)


P(2,1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

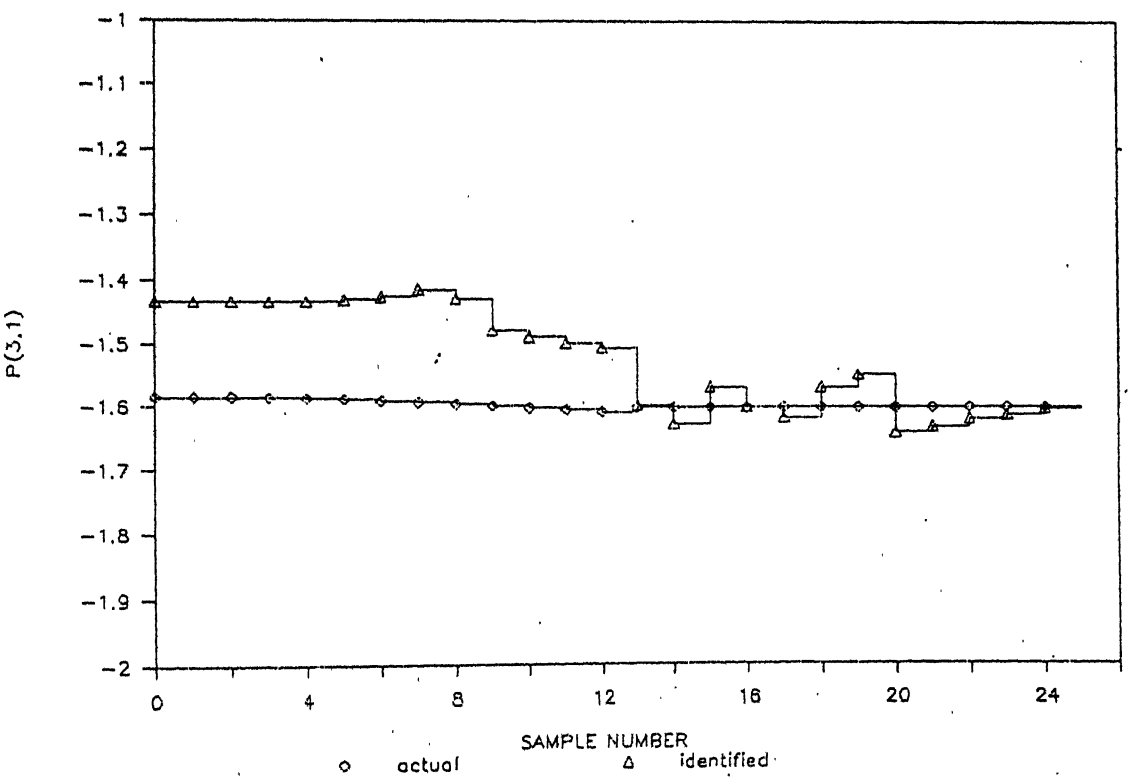
(e)


P(2,2)
0.5
0.4
0.3
0.2
0.1
0
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

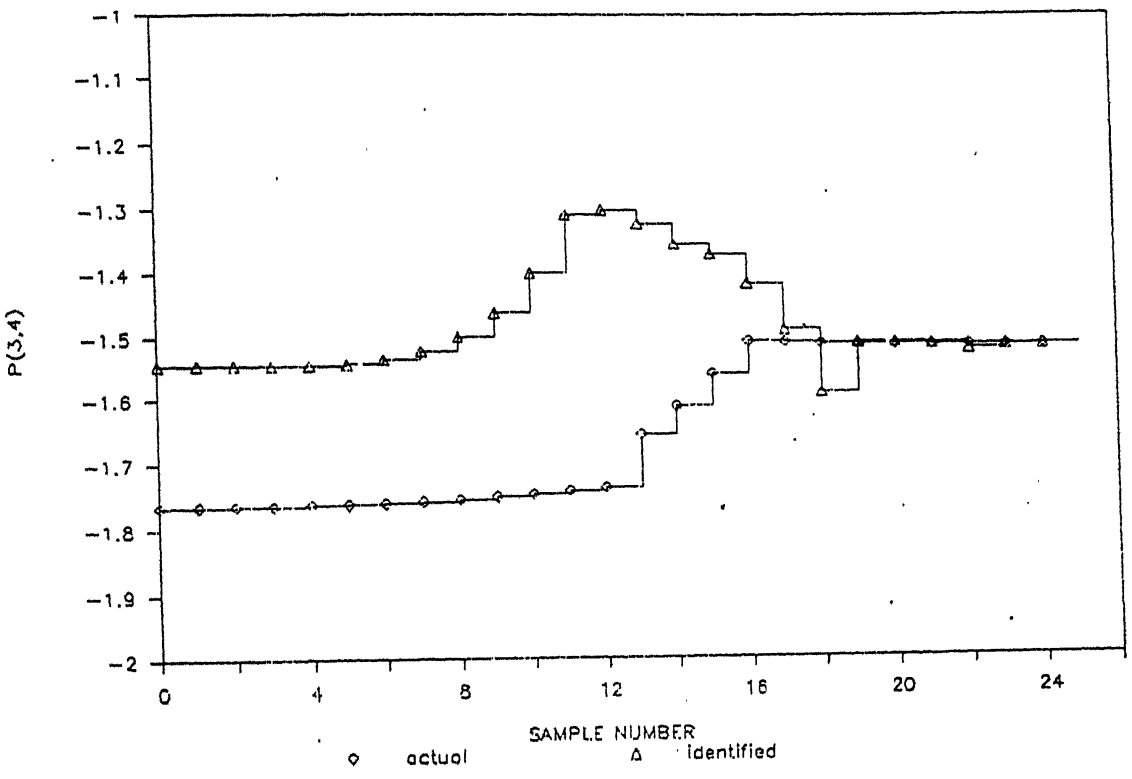
(f)


0.5
0.4
0.3
0.2
0.1
0
P(2,4)
0
4
8
12
16
20
24
SAMPLE NUMBER
actual
identified

(g)


-1
-1.1
-1.2
-1.3
-1.4
-1.5
-1.6
-1.7
-1.8
-1.9
-2
P(3,1)
0
4
8
12
16
20
24
SAMPLE NUMBER
actual
identified

(h)



(i)

Fig. 4.3: Plots Comparing Actual Parameters with Those Identified Using Model (4.31)

(a),(b),(c) - Some elements of the inertia-matrix D
(d) - An element of vector $\underline{Q}$
(e),(f),(g),(h),(i) - Some elements of matrix P

$$\begin{matrix} y_1(k) \\ y_2(k) \end{matrix} = 2I \begin{matrix} y_1(k-1) \\ y_2(k-1) \end{matrix} - \begin{matrix} y_1(k-2) \\ y_2(k-2) \end{matrix} + A' U + B'$$

$$+ \frac{C'}{T^2} \begin{matrix} [y_1(k-1) - y_1(k-2)]^2 \\ [y_1(k-1) - y_1(k-2)][y_2(k-1) - y_2(k-2)] \\ [y_2(k-1) - y_2(k-2)]^2 \end{matrix} \qquad (4.38)$$

If there is measurement error, then the measured output vector will be $\underline{Y}'(k)$ where

$$\underline{Y}'(k) = \begin{matrix} y_1(k) + e_1(k) \\ y_2(k) + e_2(k) \end{matrix} \qquad (4.39)$$

Then, eqn. (4.38) gets modified to

$$\begin{matrix} y_1(k) + e_1(k) \\ y_2(k) + e_2(k) \end{matrix} = 2I \begin{matrix} y_1(k-1) + e_1(k-1) \\ y_2(k-1) + e_2(k-1) \end{matrix}$$

$$- \begin{matrix} y_1(k-2) + e_1(k-2) \\ y_2(k-2) + e_2(k-2) \end{matrix} + A' U + B'$$

$$+ \frac{C'}{T^2} \begin{matrix} [y_1(k-1) + e_1(k-1) - y_1(k-2) - e_1(k-2)]^2 \\ [y_1(k-1) + e_1(k-1) - y_1(k-2) - e_1(k-2)] \\ [y_2(k-1) + e_2(k-1) - y_2(k-2) - e_2(k-2)] \\ [y_2(k-1) + e_2(k-1) - y_2(k-2) - e_2(k-2)]^2 \end{matrix} \qquad (4.40)$$

It must be noted that the actual output itself is not available for measurement. Hence, the regression vector used in identification itself is corrupted with measurement error. Furthermore, the error forms a moving average process and it contains nonlinear terms. It is clear that the estimated

parameters will be incorrect if identification is done based on RLS method.

Experiment nas been conducted through computer simulation to investigate the effects of measurement error. A noise of zero mean and a standard deviation of 0.001 rad. is introduced as measurement error. The results are presented in Figure 4.4. The plots of some of the actual parameters and the identified parameters in presence of measurement error are shown. The plotted elements are elements of matrix D and matrix P given by eqn. (4.36) and their identified counterparts.

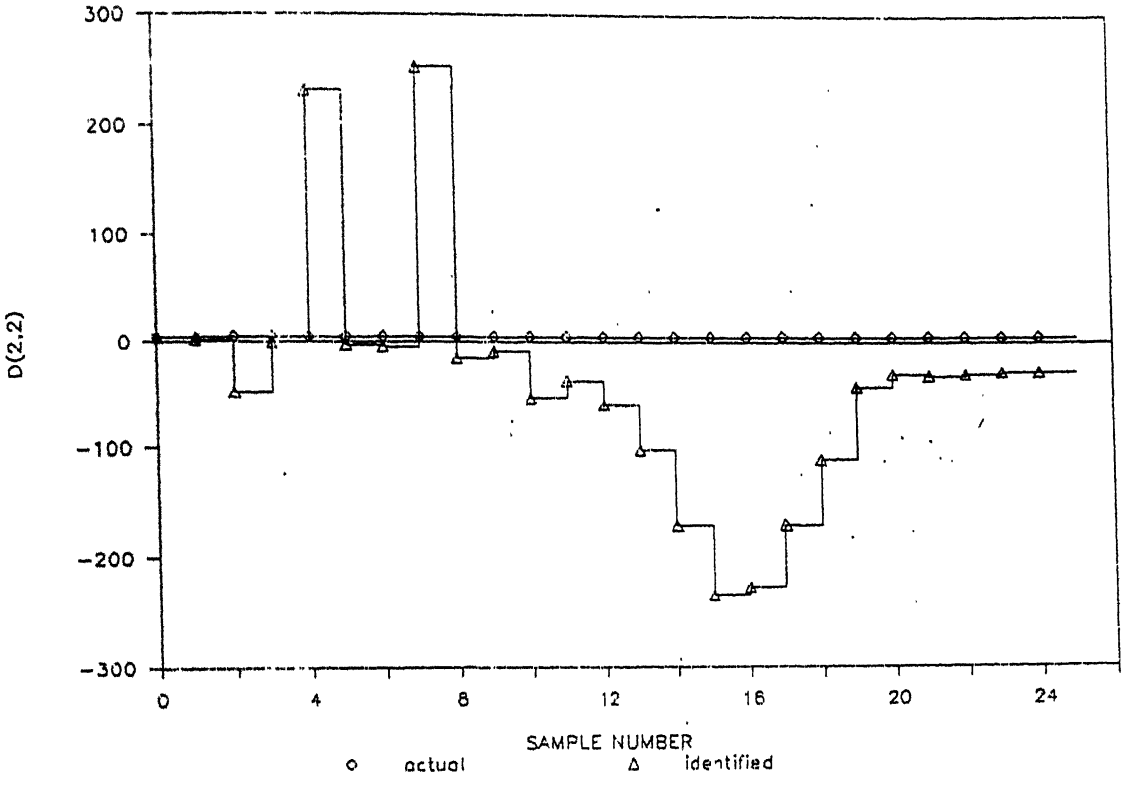
It is seen that the identified parameters do not track the actual ones. In fact, in presence of measurement error, the identifier behaves abruptly.

The measurement error analysis has been done for the model with joint positions as output. The conclusion, however, is valid for the model with velocities as output also.

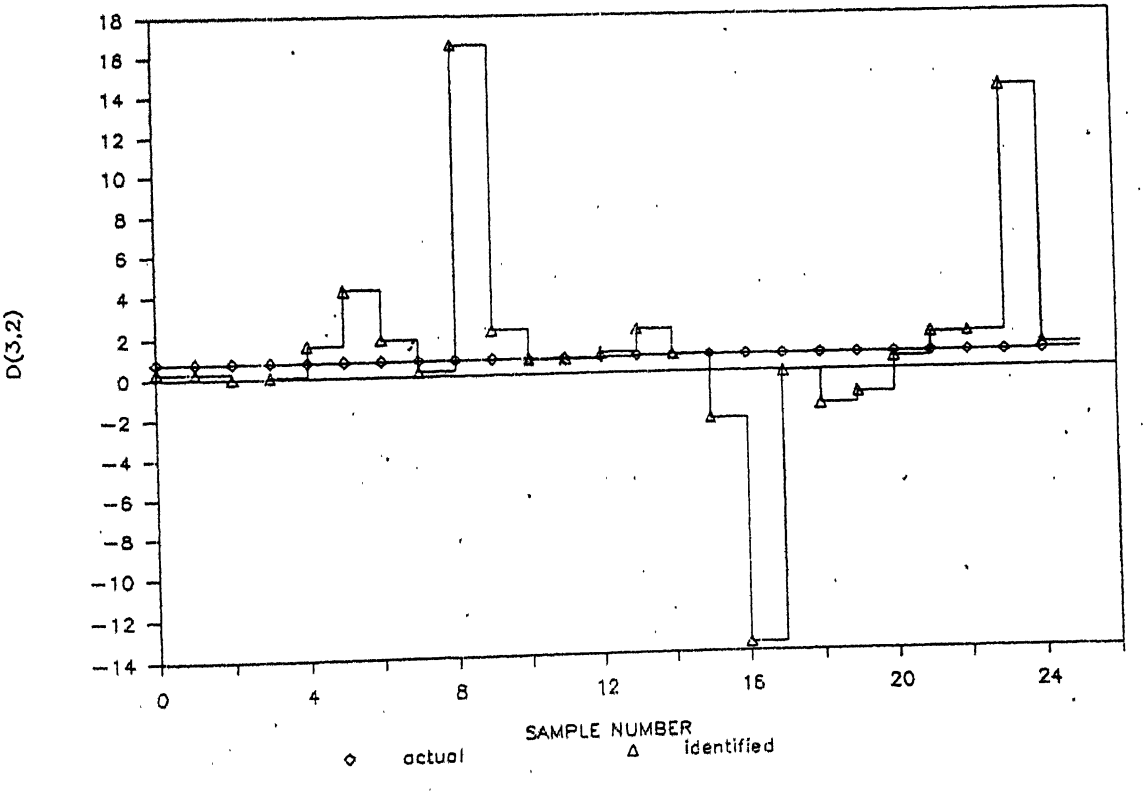
4.3.1 MODEL REFERENCE TECHNIQUE:

One way to eliminate the effects of measurement error is to use model reference technique for system identification.

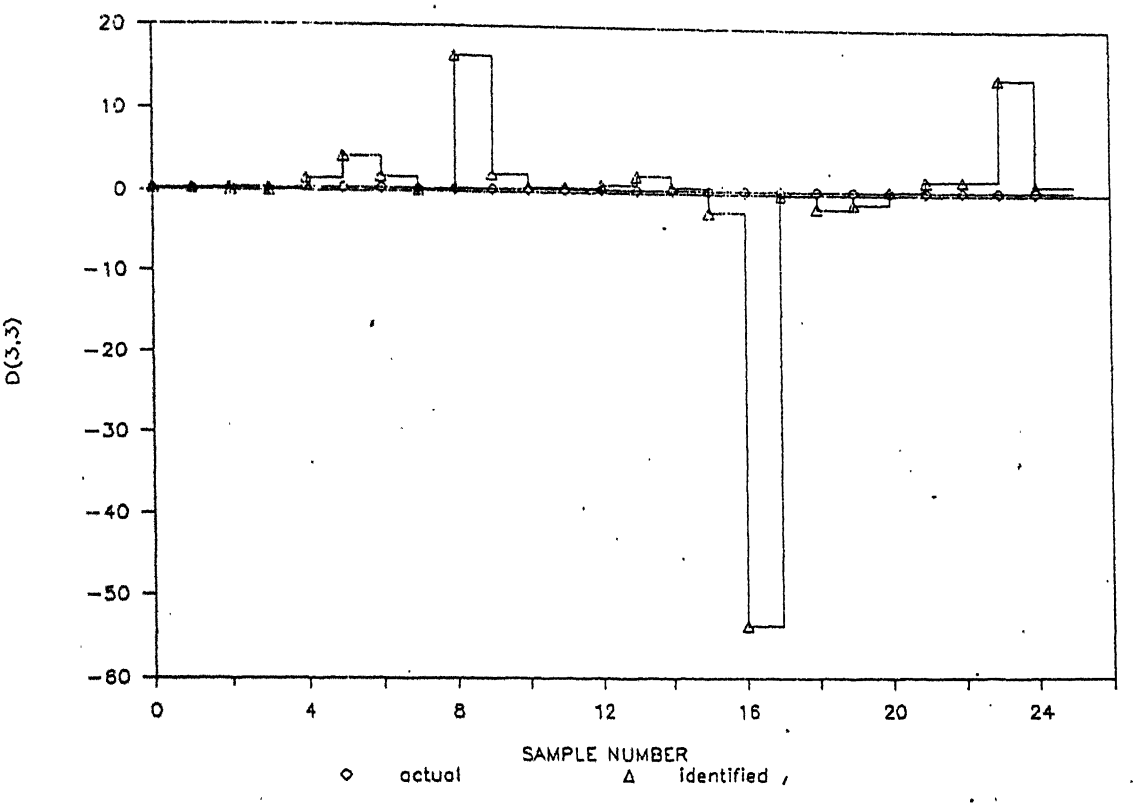
Figure 4.5 explains this approach. The method is described by Landau, 1976 . In the figure, $M(\theta)$ represents a reference model of the system. Same control input is applied to the system and the model. $\underset{\sim}{y}_m$ is the measured output of the system and y is output of the model. For identification of the system parameters, measured output from the system is compared to that of the model and model parameters are updated until the difference between the two outputs cannot be further improved. It has


300
200
100
0
−100
−200
−300
D(2,2)
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
△ identified

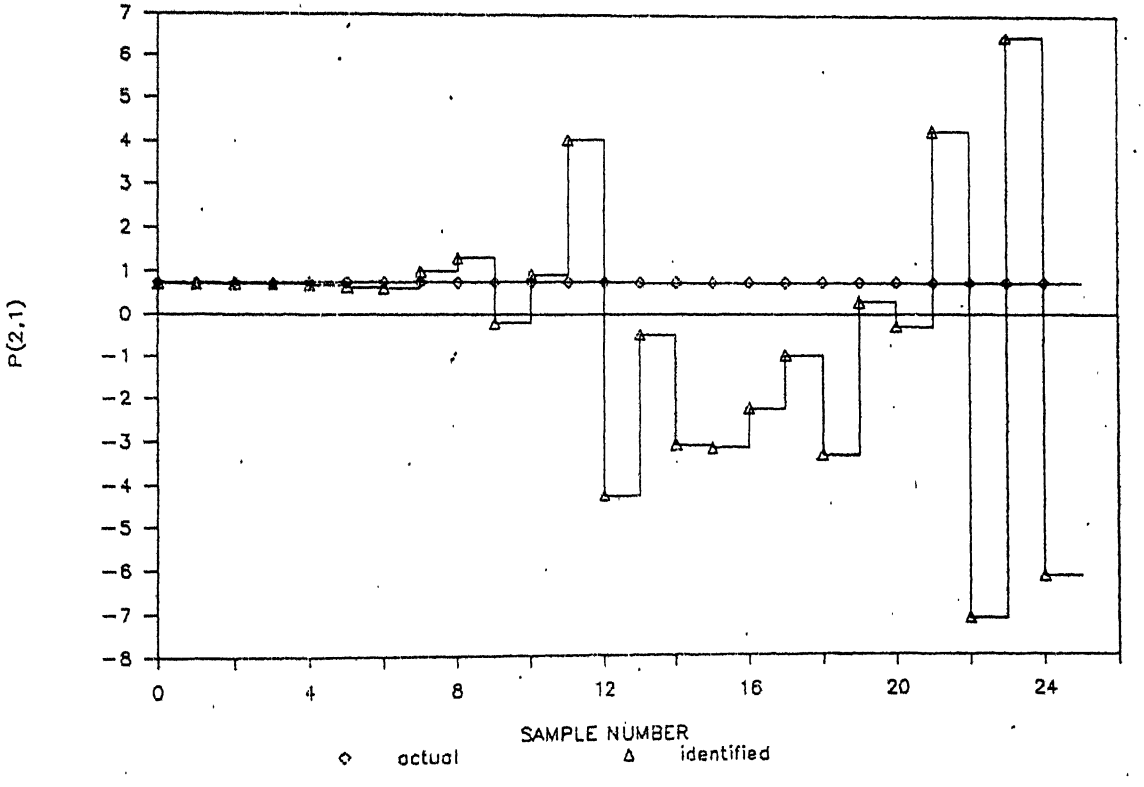
(a)


18
16
14
12
10
8
6
4
2
0
−2
−4
−6
−8
−10
−12
−14
D(3,2)
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
△ identified

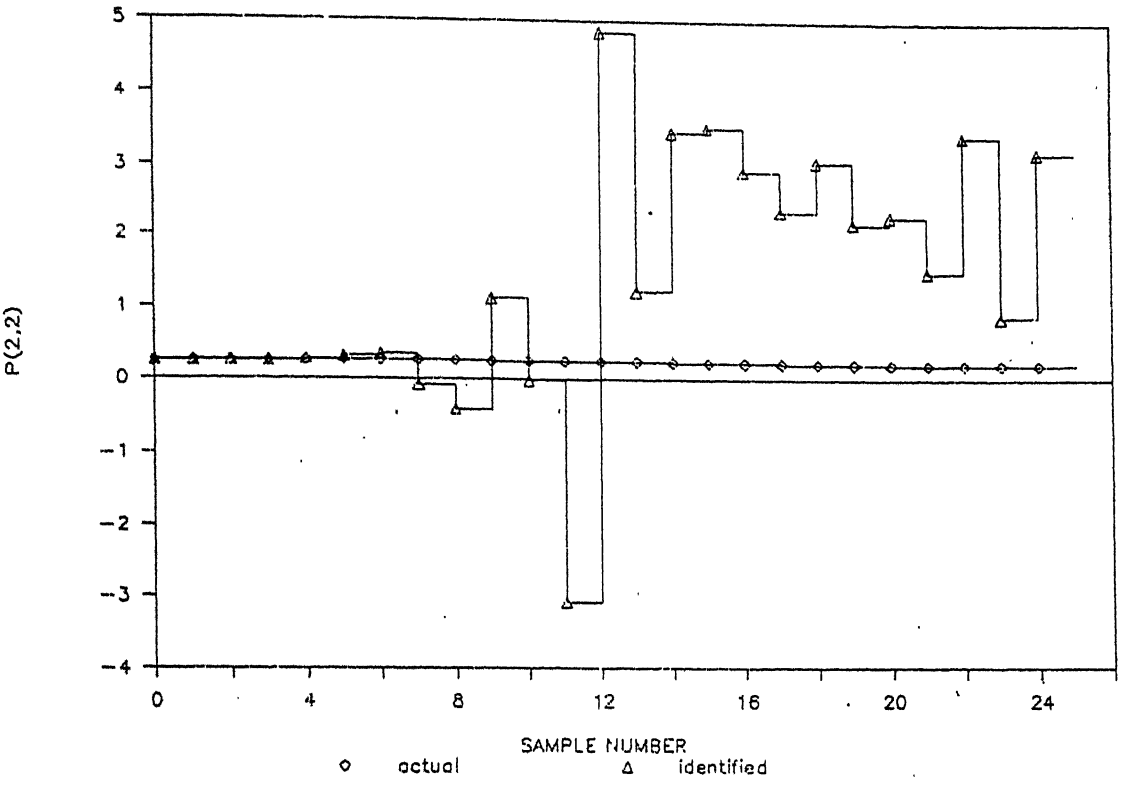
(b)


D(3,3)
20
10
0
-10
-20
-30
-40
-50
-60
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

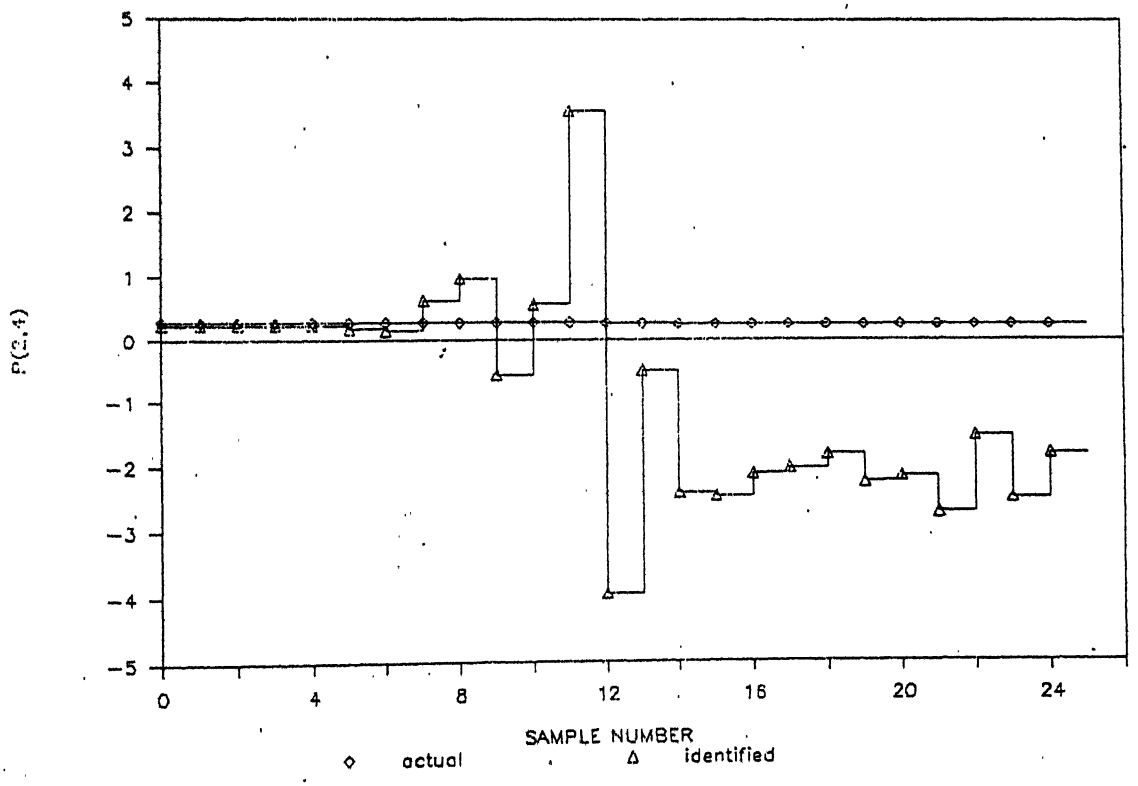
(c)


P(2,1)
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

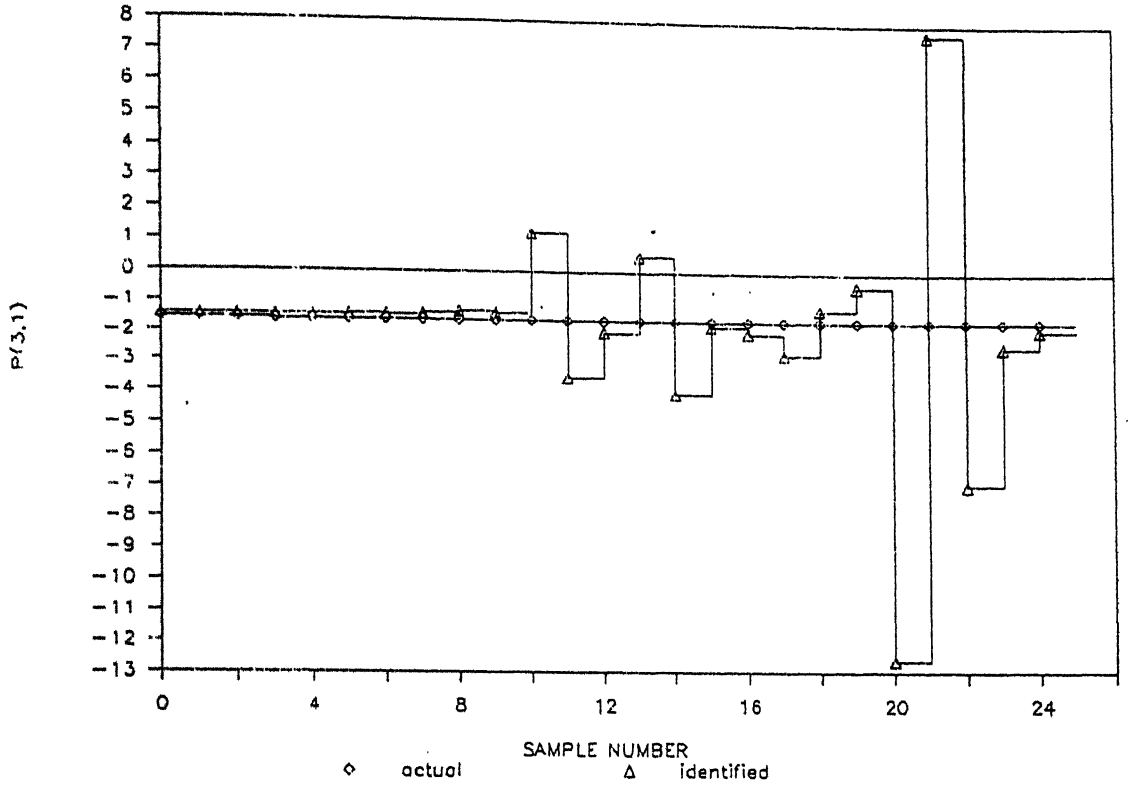
(d)


P(2,2)
5
4
3
2
1
0
−1
−2
−3
−4
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

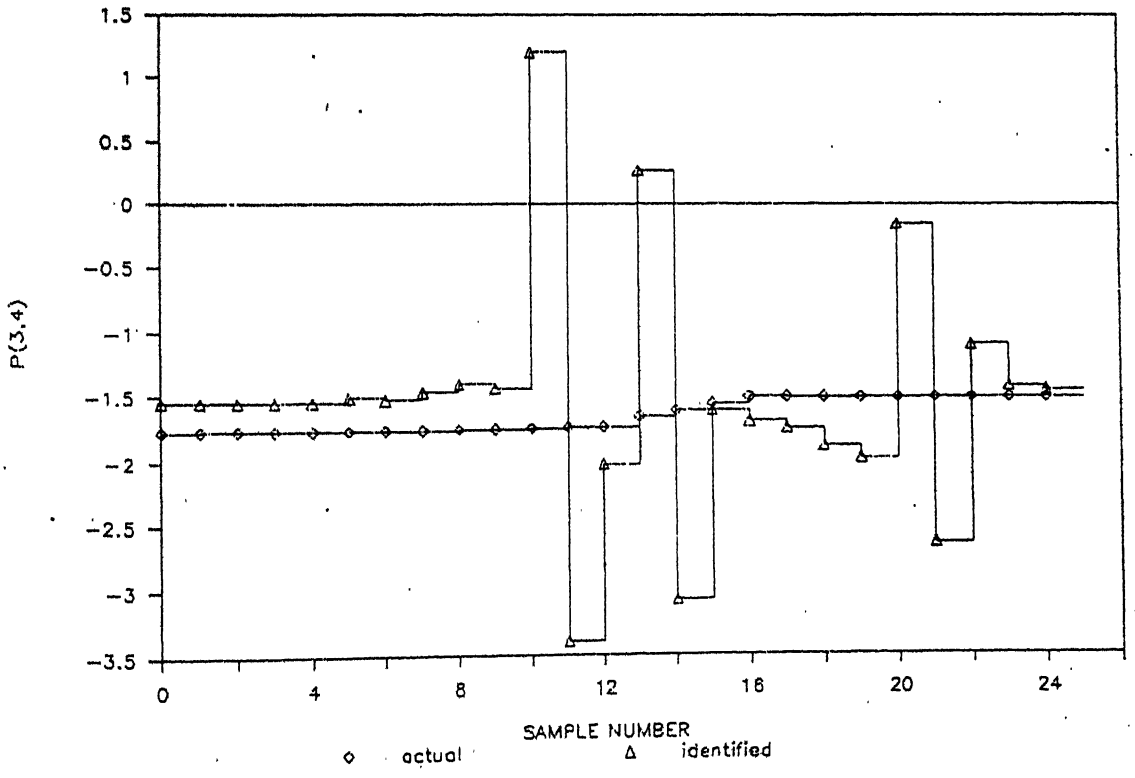
(e)


P(2,4)
5
4
3
2
1
0
−1
−2
−3
−4
−5
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

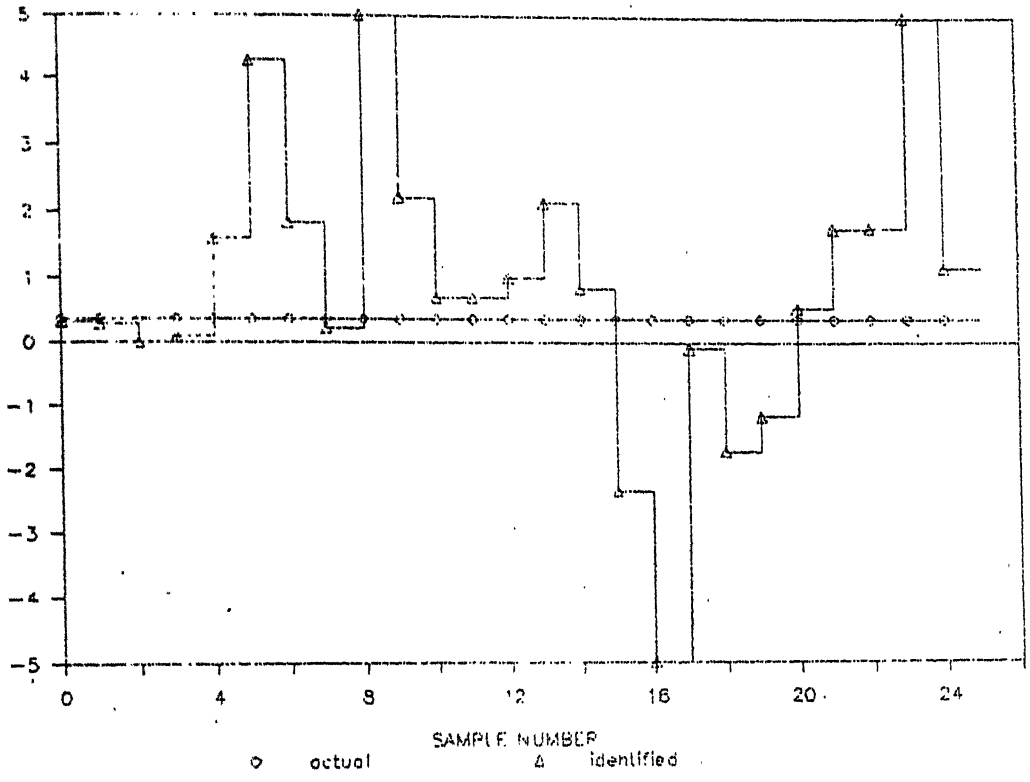
(f)


P(3,1)
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

(g)


P(3,4)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

(h)



(i)

'ig. 4.4: Plots Illustrating Effect of Measurement Noise on Identification of Parameters
(a),(b),(c) - Some elements of inertia-matrix D
(d),(e),(f),(g),(h)- Some elements of matrix P
(i) - Magnified version of plot (c)

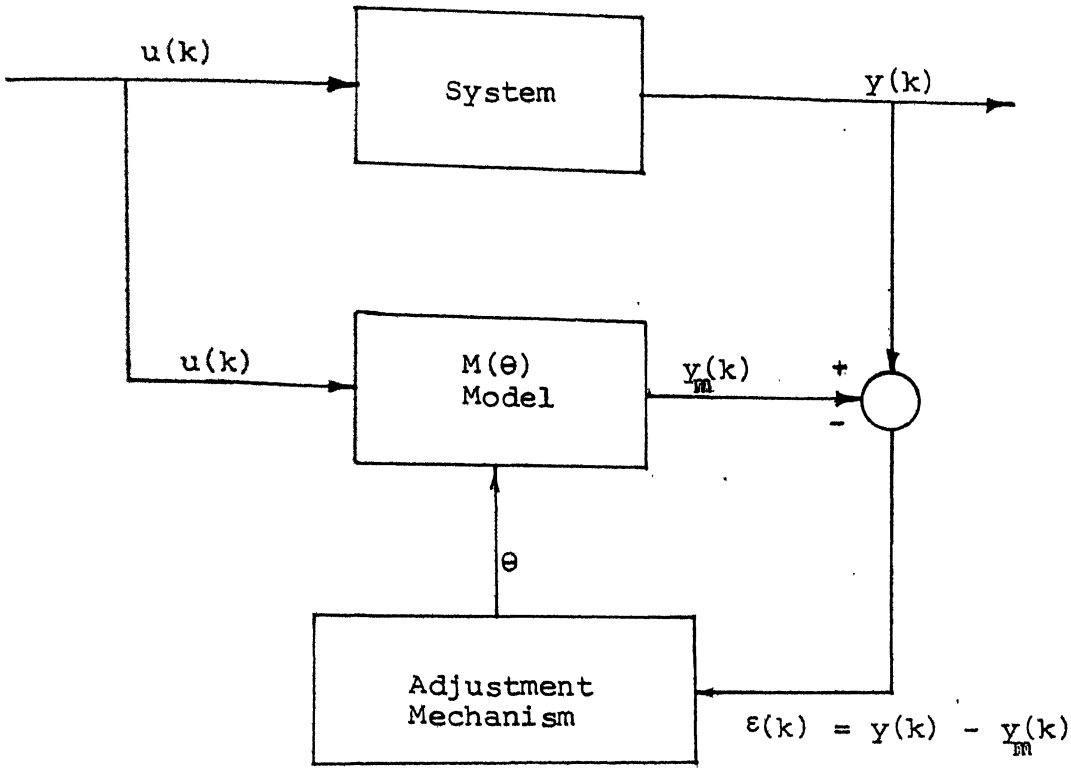


Fig. 4.5: The Model Reference Technique.

been shown that this method assures an asymptotic unbiased parameter estimation in the presence of noise obscured measurements [Landau, 1976]. This is true for single input-single output systems as well as multivariable systems.

Thus, it is seen that the RLS method do not lead to correct estimates unless the output $\underline{Y}(k)$ is directly measurable and noise-free. Otherwise the model reference technique may be used. In most of the expensive robot manipulators, the measurement of joint coordinates can be considered as noise-free. Joint coordinate sensors are most often high precision and high quality digital encoders. However, they have to be tested frequently for accuracy.

4.4 EFFECTS OF UNMODELED DYNAMICS:

If some system dynamic parameters are neglected while modeling the robot arm, it will contribute to the equation error. The effects of system dynamics are obviously non-white. The estimated parameters can then be biased and the controller is likely to fail.

In section 4.2, a model of n-jointed robot arm is derived with position as output. Comparing this model with the linearized model (3.7) obtained by Koivo, Guo, 1983, the following equations are obtained.

$a_o = B'$

$A_1 = 2I$

$A_2 = -I$

$B_1 = 0$

$B_2 = A'$

$e_k = C' \underline{v}_{k-2}$

Thus, it can be seen that model (3.7) neglects the Coriolis and centrifugal effects.

Open loop identification of PUMA-560 has been simulated using model (3.7). Since no term in robot dynamics see eqn. (3.1) involve derivative of torque input, B_1 is assumed to be zero. A_1, A_2 are initialized to 2I and -I respectively. $\underline{a}_o$ and B_2 are initialized appropriately using known values of gravity vector and inertia matrix.

Since matrix B_2 is needed to model the inertia terms, it is reasonable to expect that the matrices A_1 and A_2 will be

updated so as to account for Coriolis and centrifugal effects. Under this assumption, identified matrix B_2 will completely represent the inertia terms in eqn. (3.1).

The simulation results are presented in Figure 4.6. The plots of some of the actual parameters (elements of D matrix) and their identified counterparts are shown.

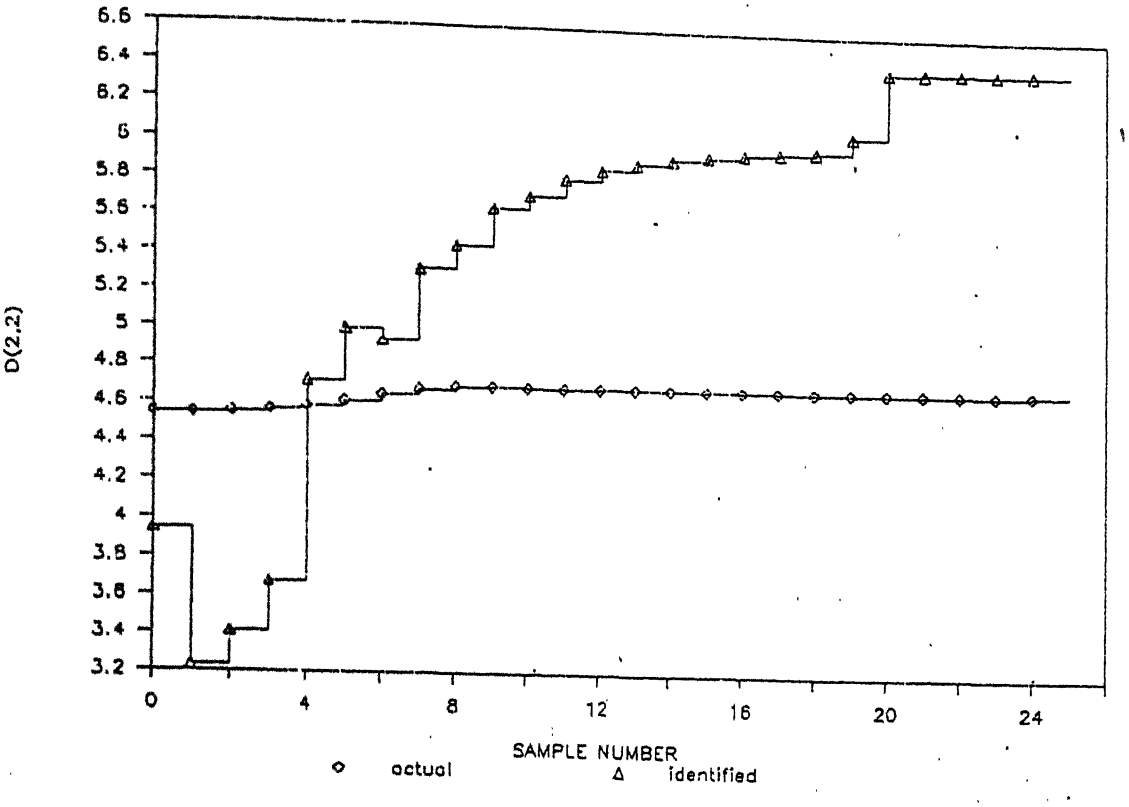
It is seen that the parameters identified using model (3.7) do not track the actual parameters. As a result, the controller based on this model may not be satisfactory.

4.5 CONCLUSION:

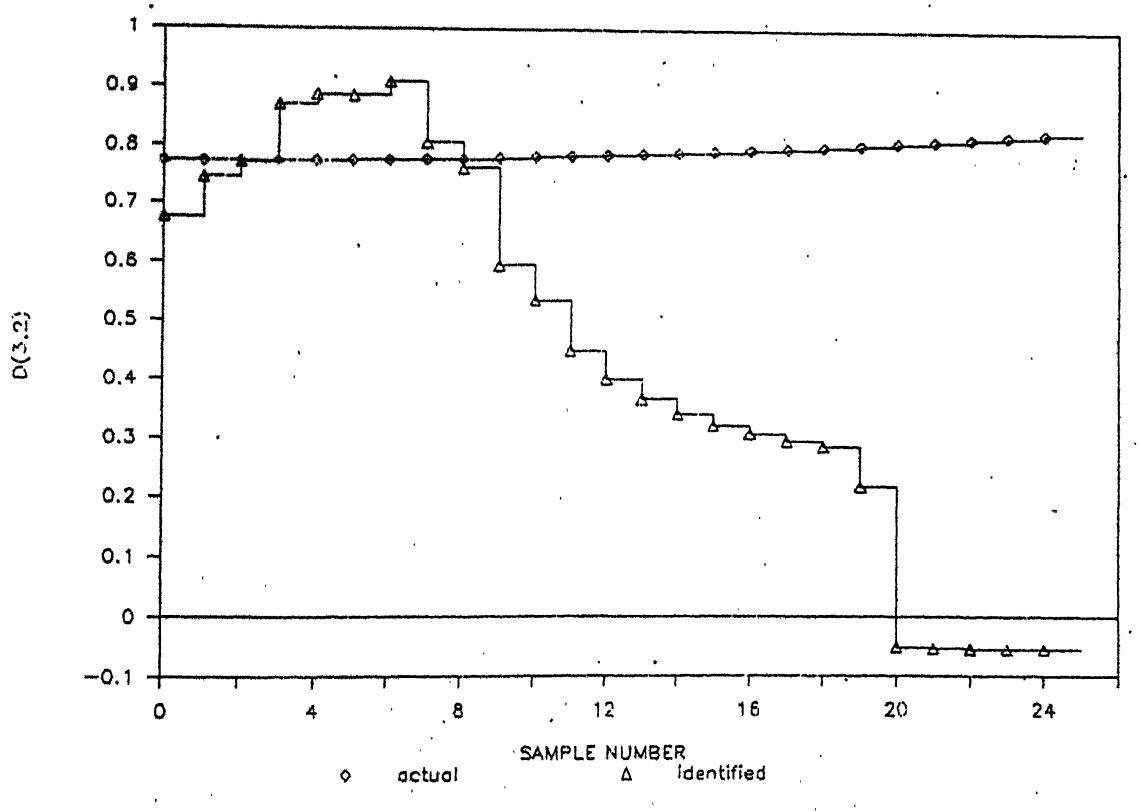
In this chapter, Computed Torque Adaptive Control has been investigated. A new model is derived which uses only joint position measurements as model output to identify the parameters. It has been shown that if the measurements are not noise-free, the parameters will not be identified correctly. In such cases, use of model reference technique is suggested.

It is also concluded that the identified parameters are going to be biased if some system dynamic parameters are neglected. However, this scheme may work in closed loop. But to what extent it will work remains to be seen.

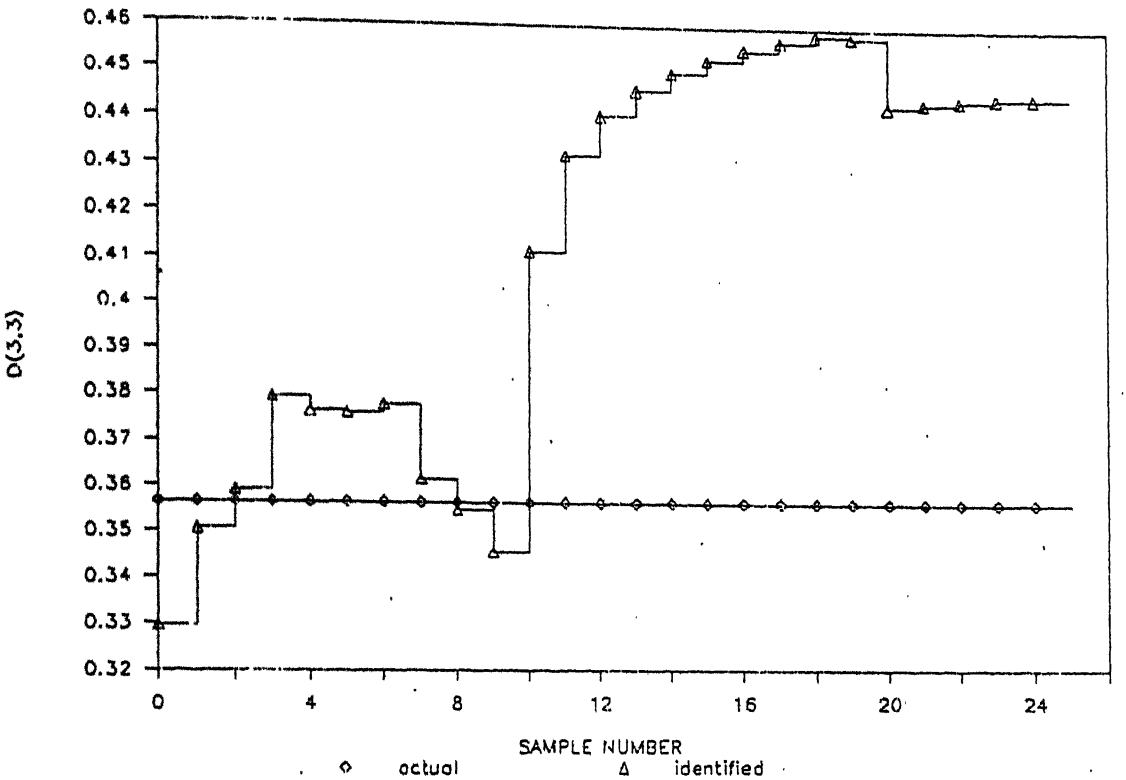
From discussion in Chapter 3 and Chapter 4, it can be seen that Computed Torque Adaptive Control is superior to all other algorithms. This algorithm is recommended for implementation. Note that, a controller based on the modified model (with position as output) will also work as good as computed-torque adaptive control.


D(2,2)
6.6
6.4
6.2
6
5.8
5.6
5.4
5.2
5
4.8
4.6
4.4
4.2
4
3.8
3.6
3.4
3.2
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

(a)


D(3,2)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
4
8
12
16
20
24
SAMPLE NUMBER
◇ actual
Δ identified

(b)



(c)

Fig. 4.5: Plots Illustrating Effect of Unmodeled Dynamics on Identification of Parameters
(a),(b),(c) - Some elements of inertia-matrix D

CHAPTER 5

IMPLEMENTATION

This chapter discusses implementation details of Adaptive Computed Torque Control Patnaik, 1987 . Identification and control algorithm is first presented from a different viewpoint so as to ease subsequent discussions. Then two ways in which multiprocessing can be achieved are proposed.

5.1 SYSTEM IDENTIFICATION:

The basic algorithm of system identification using RLS method is given by eqns. (2.3), (2.4) and (2.5). The algorithm is applied to model (4.20) where Θ and $\underline{\Psi}$ are given by eqns. (4.18) and (4.19) respectively. It must be noted that eqn. (4.20) represents a multivariable model. The parameter Θ in eqn. (4.20) is a matrix and no more a vector. Hence all the RLS equations presented in Section 2.2 have been modified appropriately.

This algorithm can be rewritten in a number of steps. The step-by-step version of the algorithm is presented for a six-joint robot by the following equations. It is to be noted that

$\underline{Y}(k)$ is a (6 x 1) vector

$\Theta(k)$ is a (34 x 6) matrix

$\underline{\Psi}(k)$ is a (34 x 1) vector

$P(k)$ is a (34 x 34) matrix.

Step 1: $\underline{e}(k) = \underline{Y}(k) - \theta^T(k-1)\,\underline{\psi}(k)$

$\underline{e}(k)$ is a (6 x 1) vector.

Step 2: $\underline{PHI} = P(k-1)\,\underline{\psi}(k)$

$\underline{PHI}$ is a (34 x 1) vector.

Step 3: $TEMP = \underline{\psi}^T(k)\,\underline{PHI}$

TEMP is a scalar.

Then,

$$TEMP = TEMP + \lambda$$

Step 4: $\underline{L}(k) = \underline{PHI}/TEMP$

$\underline{L}(k)$ is a (34 x 1) vector.

Step 5: $EL = \underline{L}(k)\,\underline{e}^T(k)$

EL is (34 x 6) matrix.

Step 6: $\theta(k) = \theta(k-1) + EL$

Step 7:i) $LPHI = \underline{L}(k)\,(\underline{PHI})^T$

LPHI is (34 x 34) matrix.

ii) $P(k) = P(k-1) - LPHI$

iii) $P(k) = P(k)/\lambda$

It is to be noted that step 7 involves (34 x 34) matrices. Taking advantage of the fact that they are symmetrical, only elements in upper or lower triangle of the matrix are computed and the rest are obtained at the cost of a requisite number of transfer operations.

Table 5.1 shows the number of computations needed in each step.

Table 5.1

	Multiplications	Additions
Step 1	204	204
Step 2	1156	1122
Step 3	34	34
Step 4	34	-
Step 5	204	-
Step 6	-	204
Step 7 - i	595	-
ii	-	595
iii	595	-
Total	2882	2159

5.2 CONTROL:

Control is computed from eqn. (4.15). In eqn. (4.15), the parameters A, B, C are involved, whereas identified parameters are A' = TA, B' = TB and C' = TC. Hence, multiplying (4.15) by T, the sampling period, following equation is obtained.

$$A' \underline{U}(k) = T \ddot{\underline{q}}_d + k_1(\dot{\underline{q}}_d - \dot{\underline{q}}) + k_2(\underline{q}_d - \underline{q}) + \underline{B}'(\underline{q}) + C' \underline{Y}_c(k) \quad (5.1)$$

This equation is used for the control computation.

The control computation involves (6 x 6) matrix inversion. Using Gaussian elimination technique for matrix inversion, approximately 500 multiplications and 500 additions are required for

total control computation. The computations may, however, be reduced by the use of pivoting technique as in Lee, Chung, 1984 .

The sampling frequency is to be maintained at 60 Hz. Thus, all the computations must be done within 16 ms so as to meet this requirement.

5.3 CHOICE OF ARCHITECTURE:

From the above discussions it is clear that a large amount of computation is involved and the need for multiprocessing is obvious. One way to tackle this problem is to use a super-mini computer which can compute at very high speed. However, this approach is not cost effective, and the use of low cost microprocessors is advocated. However, a single microprocessor will not be able to handle the job and hence more than one processor has to work together. What is desired in this case is not a general purpose parallel architecture, but a dedicated architecture that will act as a fast controller for the robotic system.

For implementations of centralized adaptive controllers, two different type of architectures can be used. They are (1) Array processing and (2) Pipelining.

5.3.1 ARRAY PROCESSING:

Considering the total number of arithmetic operations needed to be done in 16 ms and assuming typical instruction timings of microprocessors, it can be seen that about 5 to 8 processors have to be present in the array. In the identification algorithm, there are six rows in most of the matrix

computations. Hence, to use six processors for identification part is a natural choice. If six processors are used for identification and one for control computation, then it demands that one unit of computation (defined as one multiplication and one addition) is done in 25 to 30 μs. Suitable microprocessor can be selected to satisfy this requirement.

As control computations can be done within 16 ms, a separate processor can be used for this purpose. As soon as $\theta(k)$, the parameter matrix, is computed by the identifier, controller can start computations based on it. For this reason, only identifier architecture is discussed here.

The Architecture of Multiprocessing System:

The architecture of the system is shown in Figure 5.1. The array processor is a synchronous parallel computer with multiple arithmatic logic units (ALUs), called the Processing Elements (PEs), that can operate in parallel. By replication of ALUs one can achieve the spatial parallelism. The PEs are synchronized to perform the same function at the same time. Scaler and control type instructions are directly executed in Control Unit (CU). Each PE consists of an ALU with registers and a local memory.

The architecture is essentially 'Single Instruction stream-Multiple Data streme' "SIMD" like. In SIMD architecture, vector instructions are broadcast to the PEs for distributed execution over different component operands fetched directly from the local memories. Instruction fetch and decode is done

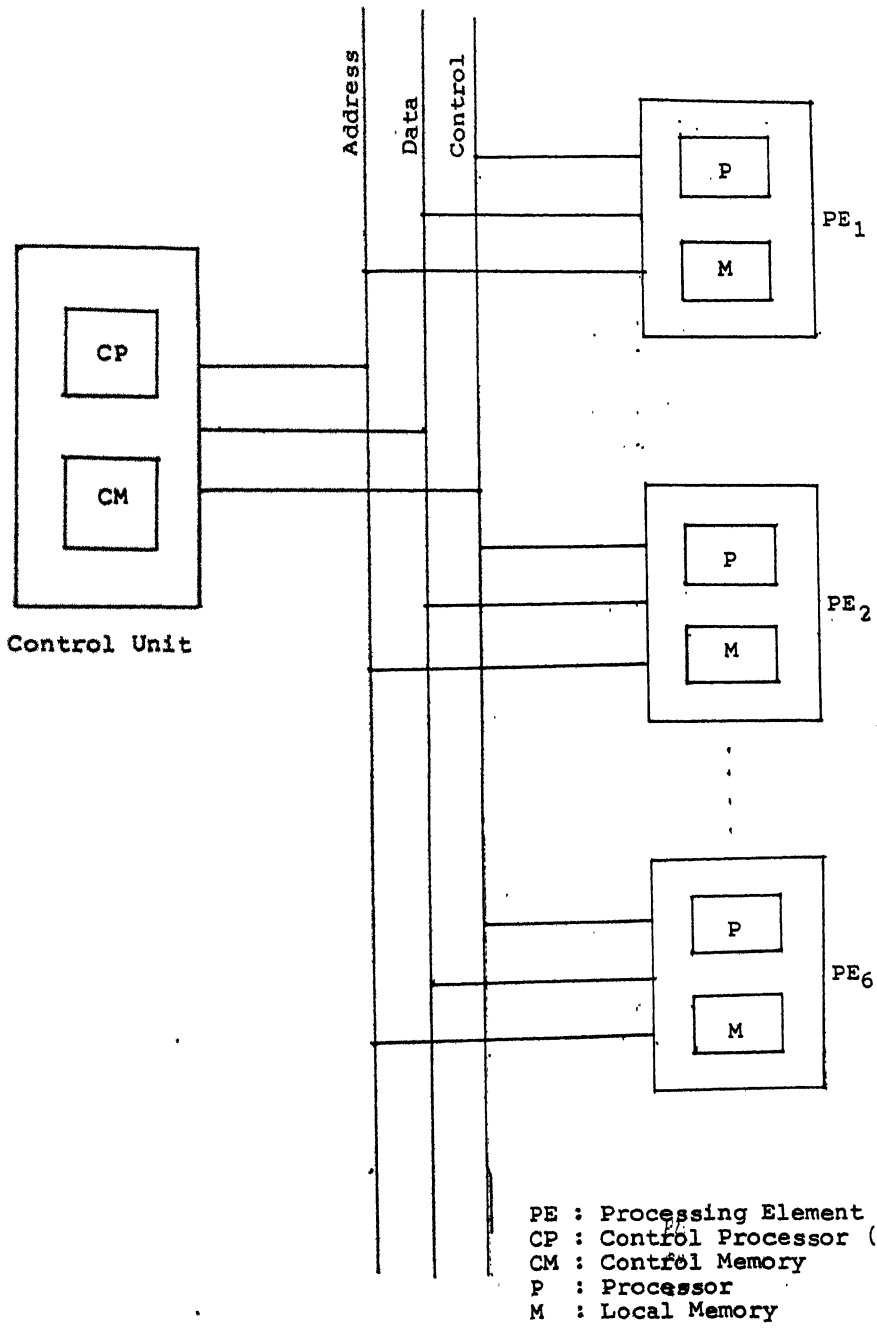


Fig. 5.1: Array Processing.

by CU. The PEs are passive devices without instruction decoding capabilities.

A slight variation of this scheme is proposed here. The steps given in Section 5.1 can be grouped into different sets. One set essentially requires that during the execution of that set each PE can work individually without talking to the CU. The actual program for each set of execution may well be replicated in each PE. The algorithm can then proceed via a series of start and stop signals between CU and the PEs. This is similar to the microprogramming concept in which one macro-instruction causes execution of a set of micro-instructions.

The saving in broadcast time, so obtained, is achieved at the cost of excess memory. It is to be noted that this optimization is possible only because it is going to be a dedicated system.

The Multiprocessing Algorithm:

Notation: Let X be a vector or a matrix. Then X_i denotes part of X that is processed by PE_i and is stored in its local memory. Then, the algorithm is optimized for parallel computation (is) as follows.

Set 1 : CU broadcasts $\Psi(k)$;

PE_i fetches $y_i(k)$;

PE_i computes $e_i(k)$; step 1 completed.

PE_i computes PHI_i ; 6 elements each

; except PE_6 which computes 4 elements, step 2

; completed.

PE_i computes $TEMP_i^T = PHI_i^T \Psi(k)$; Scalar

; product is partially computed by each PE. Each PE

; uses part of $\Psi(k)$ and computed part of PHI.

PE_i returns $TEMP_i$ to CU.

Set 2 : CU computes TEMP and broadcasts

TEMP and λ ; $TEMP = \lambda + \sum_{i}^{6} TEMP_i$

; step 3 completed

PE_i computes $L_i(k) = PHI_i/TEMP$ and

returns it to CU. ; step 4 completed.

Set 3 : CU broadcasts $L(k)$; It is

; preferred to broadcast $L(k)$ rather than $e(k)$

; for step 5. Otherwise it would demand many

; transfers of elements of EL to do steps 6 and 7

; efficiently.

PE_i computes $EL_i^T = e_i(k)\ L^T(k)$;

; step 5 completed.

PE_i computes $\theta_i^T(k) = \theta_i^T(k-1) + EL_i^T$;

; step 6 completed.

PE_i computes $LPHI_i^T = PHI_i\ L_k^T$;

; step (7)(i) completed.

Transfers of LPHI elements.

Set 4 : PE_i computes $P_i(k) = \dfrac{P_i(k-1) - LPHI_i}{\lambda}$;

; Each PE computes approximately 100 elements

; of P_i.

; Transfer of calculated elements of P to

; other triangle.

End of Recursion.

5.3.2 PIPELINING:

Pipelining offers an economical way to realize parallelism. To achieve pipelining, the input task (process) is subdivided into a sequence of subtasks, each of which can be executed by a specialized hardware stage that operates concurrently with other stages in the pipeline. Successive tasks are streamed into the pipe and get executed in an overlapped fashion at the subtask level.

Figure 5.2 explains this approach. With respect to the figure, each module in the architecture does the required computations in 16 ms. Each module is either a Von Neumann machine (a single CPU) or an array processor as described in the previous section.

Referring back to the step-by-step algorithm given in Section 5.1, Step 1 is done by one processor. Step 2 is done by two (or at the most three) processors. It starts concurrently with Step 1. Steps 3, 4, 5 and 6 are done by one processor. Step 7 is done by two (or at the most three) processors. It starts concurrently with Step 5.

How many processors one should use per module primarily depends upon the computation speed of the available processors.

One observation must be made here. The control to be applied at k will be computed using the identified parameters of the robot a few (say, N) samples before. This phenomenon is explained in Figure 5.3. In the pipeline architecture proposed, N is equal to 3.

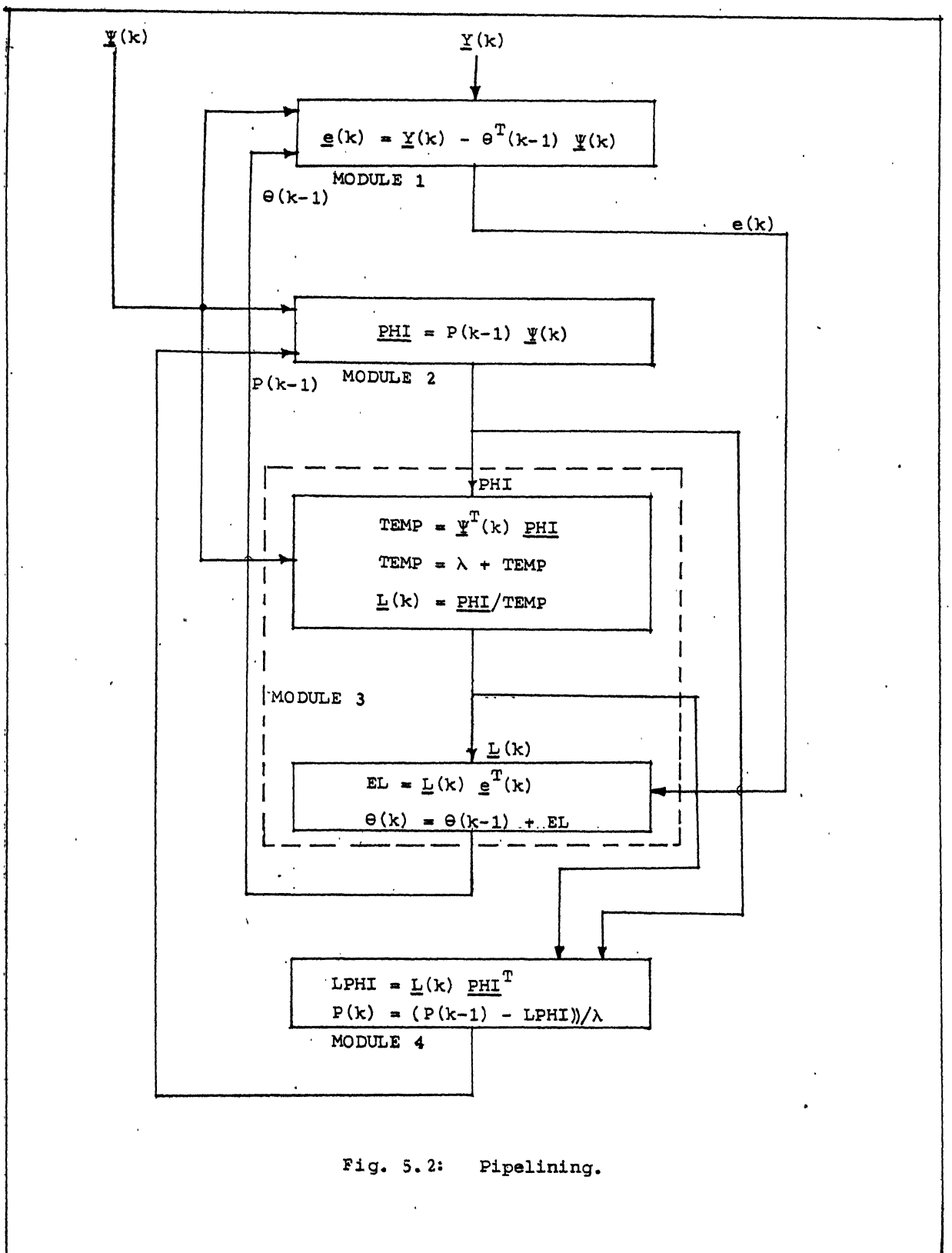


Fig. 5.2: Pipelining.

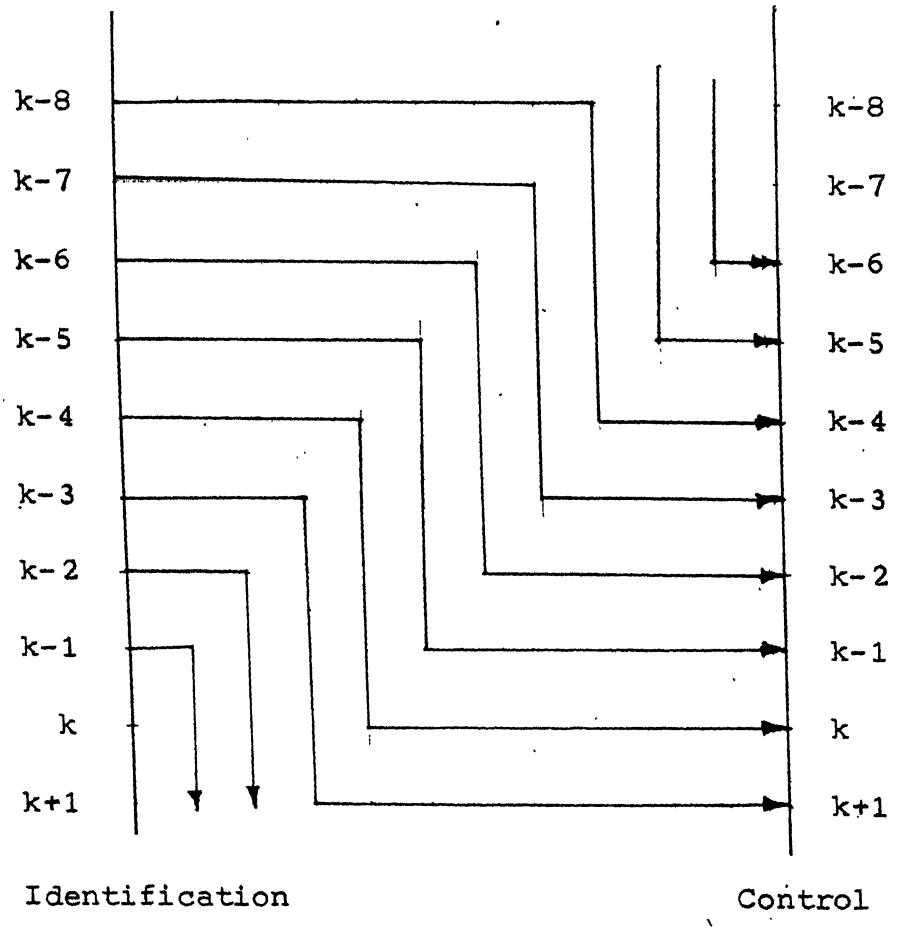
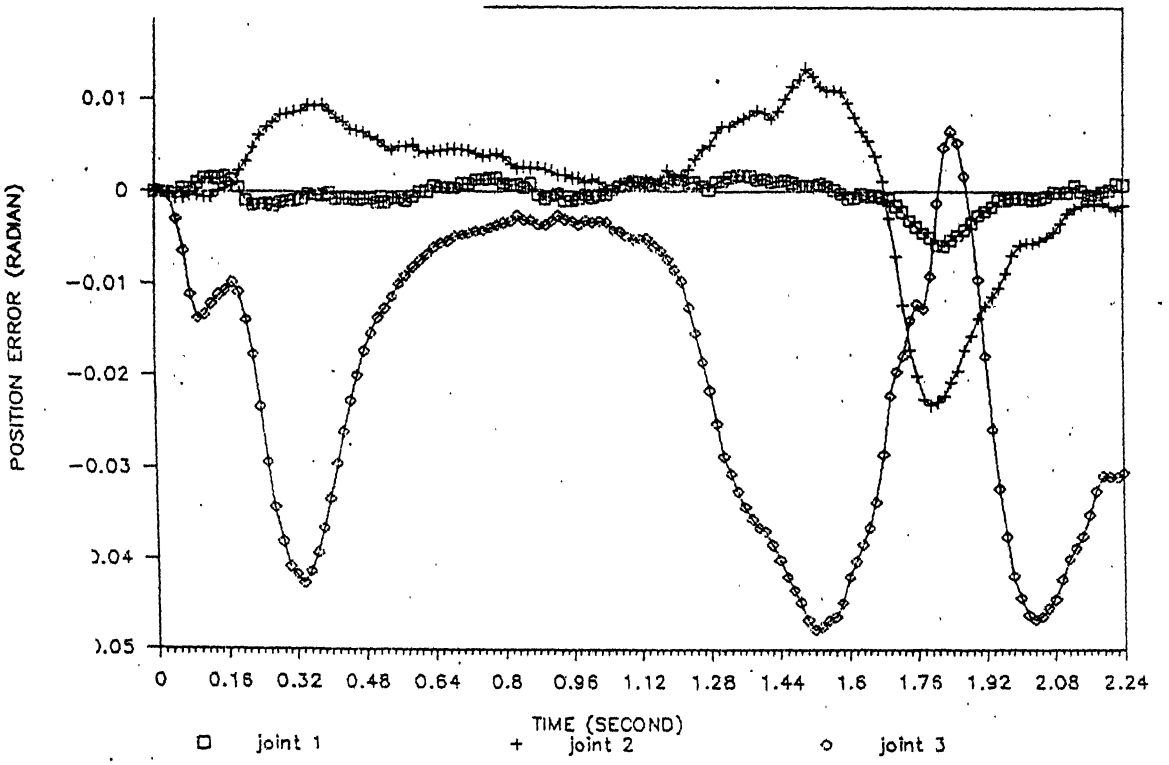


Fig. 5.3 Pipelining with N = 4.

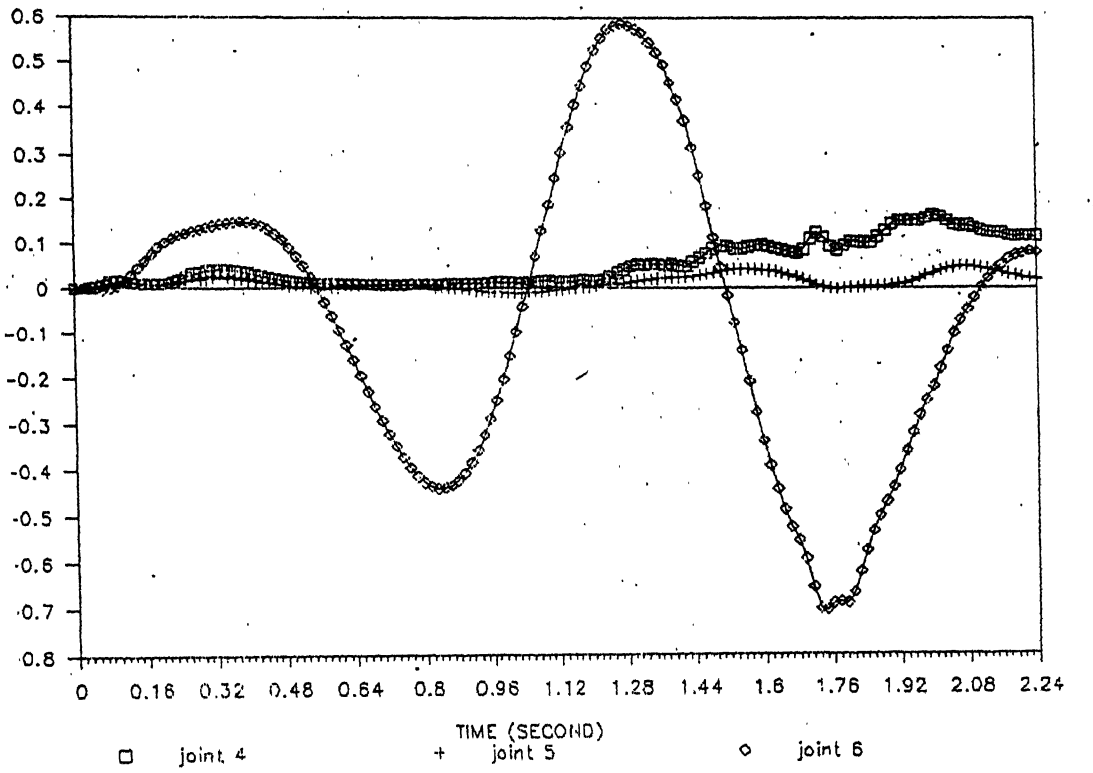
This phenomenon has been simulated on DEC-1090. Experiments have been conducted on PUMA-600 through simulation for N = 3, 4, 5, 6, 7.

With N = 3, 4, 5 satisfactory tracking has been obtained for the trajectories in which the end effector moves with a linear speed upto 1.25 m/s. As N increases, the tracking accuracy degrades, and at N = 6 and 7 it fails to track the desired trajectory. Figure 5.4 shows the tracking performance for N = 3

Experiments with variation in the speed of tracking show that at high speeds (above 3.5 m/s) tracking accuracy degrades even with N = 2. This obviously restricts use of pipelining at high speeds.



(a)



(b)

5.4: Tracking Error with Pipelining
(a) First three joints
(b) Last three joints

The robot parameters vary with time. As N increases, the controller gets the parameters which are farther away from actual parameters. The same is true if the speed is increased. Of course, if the parameters of robot change very slowly in a trajectory, the performance will not be significantly affected by N.

It must be noted that putting Adaptive Computed-Torque Control algorithm in the architecture proposed, changes the algorithm itself. For example, the equations $\underline{e}(k) = \underline{y}(k) - \Theta^T(k-1)\ \underline{\Psi}(k)$ and $\underline{PHI} = P(k-1)\ \underline{\Psi}(k)$ now become $\underline{e}(k) = \underline{y}(k) - \Theta^I(k-3)\ \underline{\Psi}(k)$ and $\underline{PHI} = P(k-3)\ \underline{\Psi}(k)$ respectively. This is equivalent to introducing a delay in the feedback path of the closed loop system. The experiments show that the closed loop can accommodate this delay to some extent.

Figure 5.3 shows computations of the modified algorithm that would be done at instant k. It is to be noted that Figure 5.2 is no more applicable as can be seen in the previous paragraph.

From the above discussion one may feel that, a scheme, in which the controller parameters are calculated each sample instant, whereas the identifier updates the parameters after each N samples will also work. At each sample instant, the controller takes the up-to-date measurements but uses old parameter values. Here, the sampling frequency used by the controller is still 60 Hz which is well above natural resonant frequency of the robot. This will reduce the number of processors to be used in the pipeline.

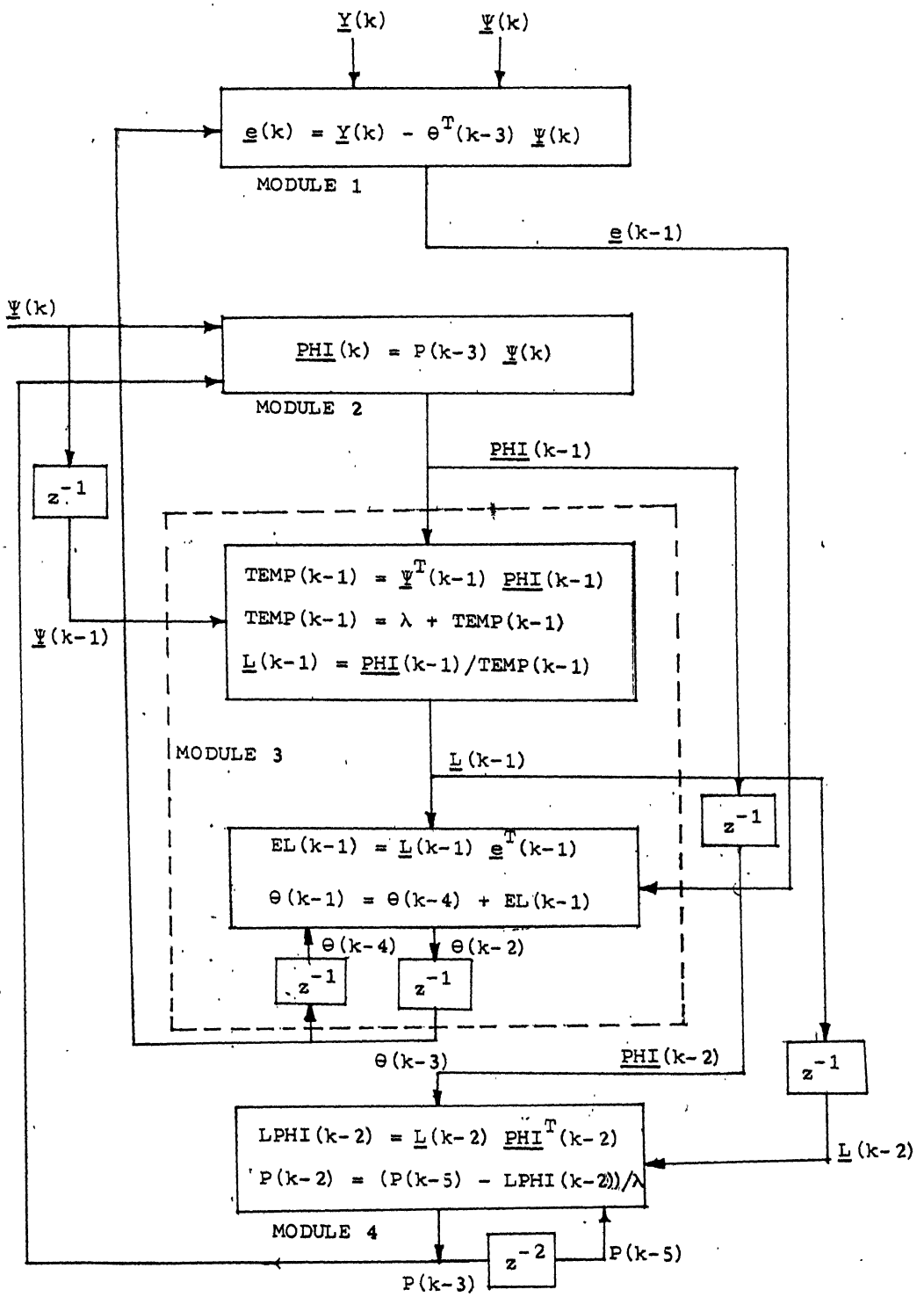


Fig. 5.5: Pipelining : Computations at k.

Though this seems highly desirable, it must be noted that this is equivalent to the reduction of sampling rate of the identifier by a factor of N. In experiments through simulation, it turned out to be unsuccessful.

A similar experience is also noted in Koivo, Guo, 1983 where the authors suggested that the choice of the sampling rate should be no less than 60 Hz to achieve a sufficiently smooth control.

5.3.3 COMPARISON OF ARRAY PROCESSING AND PIPELINING:

Clearly, pipelining should not be used where fast arm movements are involved. The method will fail for the reasons discussed. However, for low and medium speed operations, pipelined architecture has many desirable features.

The design of multiprocessor system is modular in this case. If a module fails to give satisfactory data processing speed, it can always be augmented with one more processor. In array processor, whole parallel algorithm will have to be rescheduled.

In array processing, one has to resort to very fast processors such that the condition of using six PEs is met. One can easily think of the complications arising in architecture and algorithm if one adds one more PE.

If the control algorithm is modified in future, the change can be easily implemented in pipelined architecture, where as array processing architecture might have to go through major reshuffle.

The pipelined architecture can thus be attractive for low and medium speed applications. However, array processing architecture is recommended here since it gives better performance in terms of speed and tracking accuracy, though it is likely to be costlier and complex in terms of design.

CHAPTER 6

CONCLUSIONS

In this thesis, the robot control problem is studied from various angles. Some of the conclusions drawn from the study are presented here. The discussion is concluded with a few suggestions for further work.

6.1 GENERAL CONCLUSIONS:

From the discussion and simulation studies in the previous chapters, the following general comments can be made.

(1) The non-adaptive controllers are not suitable for wide range of manipulator motions and payloads.

(2) The decentralized approach to robot arm adaptive control treats each joint of the robot arm as a simple joint servo-mechanism. The approach neglects the coupling forces between the joints which may be severe for manipulators with rotary joints.

(3) Performance of the controllers depend significantly on values to which control parameters are initialized. In most of the algorithms, initialization has to be done by trial and error. It is also difficult to figure out the correlations of these values with overall performance of the controller.

(4) Linearized models, in general, are not guaranteed to show minimum phase behavior. Minimum phase requirement is satisfied only if the manipulator path is well planned. However, non-minimum phase behavior can be handled with

LQG control or with pole-placement. This again involves careful selection of control polynomials.

(5) Computed Torque Adaptive Controller is most robust of the algorithms discussed in terms of stability and tracking accuracy. In this algorithm, the parameter matrix can be initialized in the easiest way using initial position of the robot arm.

(6) A model is derived which utilizes only position measurements. It is observed that system parameters can be identified using this model. Any standard control approach can then be applied.

(7) If there is measurement error, the robot control algorithm fails if RLS method is used to estimate the parameters.

(8) If some system dynamic parameters are neglected from the model, identified parameters are biased.

(9) Two dedicated computer architectures have been proposed to implement Computed Torque Adaptive Control algorithm. Pipelined architecture introduces delay in the feedback path of the closed-loop system. For better performance, array processor is recommended.

6.2 SCOPE FOR FURTHER WORK:

The work in the thesis is digital computer simulation oriented. However, need for more theoretical work is emphasized here. Stability and convergence of the robot control algorithms may be analysed in more details. But it must be cautioned that in this problem, it is very difficult to prove stability of the overall system.

The Computed Torque Adaptive Control gives good results at least in computer simulation. It may be implemented using proposed array processing architecture.

REFERENCES

(1) Albus J-S., 'A new approach to manipulator control: The cerebellar model articulation controller (CMAC)', J. Dynamic Syst., Meas., Contr. (ASME), pp. 220-227, Sept. 1975.

(2) Asada H., Slotine J.J.E., 'Robot Analysis and Control', John Wiley, New York, 1986.

(3) Astrom K.J., 'Introduction to Stochastic Control Theory', Academic Press, New York, 1970.

(4) Astrom K.J., Wittenmark B., 'On self-tuning regulators', Automatica, Vol. 9, pp. 185-199, 1973.

(5) Astrom K.J., Borisson U., Ljung L., Wittenmark B., 'Theory and applications of self-tuning regulators', Automatica, Vol. 13, pp. 457-476, 1977.

(6) Astrom K.J., Wittenmark B., 'Self-tuning controllers based on pole-zero placement', IEE Proceedings, Vol. 127, May 198

(7) Choi Y.K., Chung M.J., Bien Z., Lyou J., 'An adaptive regulation scheme for manipulators', Proceedings of '85 I.C.A.R pp. 259-266, Sept. 1985.

(8) Clark D.W., Gawthrop P.J., 'Self-tuning controllers', Proc. IEE, pp. 929-934, 1975.

(9) Dubowsky S., DesForges D.T., 'The application of model-referenced adaptive control to robotic manipulators', J. Dynamic Syst., Meas., Contr. (ASME), Vol. 101, pp. 193-200, Sept. 1979.

(10) Fu K.S., Gonzalez R.C., Lee C.S.G., 'Robotics: Control, Sensing, Vision and Intelligence', McGraw-Hill Book Company, 1987.

(11) Ghosh A., Ledwich G., Malik O.P., Hope G.S., 'Power System Stabilizer based on adaptive control technique', IEEE Trans. on Power Apparatus and Systems, Vol. PAS-103, No. 8, Aug. 1984.

(12) Hollerbach J.M., 'A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity', IEEE Trans. SMC-10, pp. 730-736, 1980.

(13) Horowitz R., Tomizuka R., 'An adaptive control scheme for mechanical manipulators - Compensation of nonlinearity and decoupling control', J. Dynamic Syst., Meas., Contr. (ASME) June 1986.

(14) Hwang K., Briggs F.A., 'Computer Architecture and Parallel Processing', McGraw-Hill Book Company, 1985.

(15) Isermann R., 'Digital Control Systems', Springer-Verlag, Berlin, 1981.

(16) Kahn M.E., Roth B., 'The near-minimum time control of open-loop articulated kinematic chains', J. Dynamic Syst., Meas., Contr. (ASME), pp. 164-172, Sept. 1971.

(17) Koivo A.J., Guo T.H., 'Adaptive linear controller for robotic manipulators', IEEE Trans. Automat. Contr., Vol. AC-28, pp. 162-171, Feb. 1983.

(18) Koivo A.J., 'Self-tuning manipulator control of robotic systems', J. Dynamic Syst., Meas., Contr. (ASME), Vol. 107, No. 4, Dec. 1985.

(19) Landau I.D., 'Unbiased recursive identification using model reference techniques', IEEE Trans. Automat. Contr., Vol. AC-21, pp. 194-202, 1976.

(20) Lee C.S.G., Chung M.J., 'An Adaptive Control Strategy for Computer Based Manipulators', The University of Michigan, Ann Arbor, Technical Report RDS-TR-10-82, Mar. 1982.

(21) Lee C.S.G., Chung M.J., 'An adaptive control strategy for mechanical manipulators', IEEE Trans. Automat. Contr., Vol. AC-29, pp. 837-840, 1984.

(22) Lee C.S.G., Lee B.H., 'Resolved Motion Adaptive Control for Mechanical Manipulators', The University of Michigan, Ann Arbor, Technical Report RSD-TR-10-83, Mar. 1984.

(23) Leininger G.G., Wang S.Y., 'Pole placement self-tuning control of manipulators', IFAC Symp. Computer Aided Design of Multivariable Technological Systems, pp. 27-33, 1982.

(24) Leininger G.G., Backes P.G., Chung C.H., 'Joint self-tuning with Cartesian setpoints', J. Dynamic Syst., Meas., Contr. (ASME), pp. 146-150, June 1986.

(25) Lewis F.L., 'Optimal Estimation: With an Introduction to Stochastic Control Theory', John Wiley, New York, 1986.

(26) Liu M., 'An adaptive control scheme for robotic manipulators', Proceedings of 15th International Symposium on Industrial Robots, Tokyo, Japan, Vol. 2, Sept. 1985.

(27) Liu C., Chen Y., 'Multi-microprocessor based Cartesian-space control techniques for a mechanical manipulator', IEEE J. Robotics and Automation, Vol. RA-2, No. 2, pp. 110-115, June 1986.

(28) Ljung L., Soderstrom T., 'Theory and Practice of Recursive Identification', MIT Press, Cambridge, Mass., 1981.

(29) Luh J.Y.S., 'Conventional controller design for industrial robot', IEEE Trans. Syst. Man and Cybern., Vol. SMC-13, pp. 298-316, 1983.

(30) Luh J.Y.S., Walker M.W., Paul R.P.C., 'On-line computational scheme for mechanical manipulators', J. Dynamic Syst Meas., Contr. (ASME), Vol. 102, pp. 69-76, June 1980.

(31) Luh J.Y.S., Walker M.W., Paul R.P.C., 'Resolved acceleration control of mechanical manipulators', IEEE Trans. Automat. Contr., Vol. AC-25, No. 3, pp. 469-474, June 1980.

(32) Luh J.Y.S., Lin, C.S., 'Scheduling of parallel computation for a computer-controlled mechanical manipulator', IEEE Trans. Syst., Man and Cybern., Vol. SMC-12, pp. 214-234, Mar. 1982.

(33) Ortega R., Kelly R., 'PID self-tuners: Some theoretical and practical aspects', IEEE Trans. on Industrial Electronics, Vol. IE-31, No. 4, Nov. 1984.

(34) Patnaik B.R., 'Computed Torque Adaptive Control of Robot Manipulators', M.Tech. thesis, Dept. of Electrical Engineering, IIT Kanpur, 1987.

(35) Paul R.P.C., 'Robot Manipulators: Mathematics, Programming and Control', MIT Press, Cambridge, Mass., 1981.

(36) Paul R.P.C., Wu C.H., 'Resolved motion force control of robot manipulator', IEEE Trans. Syst. Man and Cybern., Vol. SMC-12, No. 3, pp. 266-275, June 1982.

(37) Takegaki M., Arimoto S., 'A new feedback method for dynamic control of manipulators', J. Dynamic Syst. Meas., Contr. (ASME), Vol. 103, pp. 119-125, 1981.

(38) Takegaki M., Arimoto S., 'An adaptive trajectory control of manipulators', Int. J. Control., Vol. 34, pp. 219-230, 1981.

(39) Vukobratovic M., Stokic D., 'Contribution to suboptimal control of manipulator robots', IEEE Trans. Automat. Contr., Vol. AC-28, pp. 981-985, 1983.

(40) Vukobratovic M., Stokic D., Kircanski N., 'Scientific Fundamentals of Robotics 5-Non-Adaptive and Adaptive Control of Manipulation Robots', Springer-Verlag, Berlin, 1985.

(41) Whitney D.E., 'Resolved motion rate control of manipulators and human prostheses', IEEE Trans. Man-Machine Systems, Vol MMS-10, No. 2, pp. 47-53, 1969.

(42) Young K.K.D., 'Controller design for a manipulator using theory of variable

APPENDIX I

LINK PARAMETERS AND MASS PROPERTIES OF PUMA 560 ROBOT

The link parameters associated with the first three joints of the PUMA 560 robot arm are given below in a tabular form

Joint number (i)	q_i in degrees	α_i in degrees	a_i in meters	d_i in meters
1	-160 to +160	-90	0	0
2	-225 to +45	0	0.432	0.1495
3	-45 to +225	90	0	0.0

The mass properties are specified as follows:

I_i is the inertia matrix for i^{th} joint. Then

$$I_i = \begin{bmatrix} I_{ixx} & 0 & 0 \\ 0 & I_{iyy} & 0 \\ 0 & 0 & I_{izz} \end{bmatrix}$$

diag. I_1 = 0.0071, 0.0267, 0.0267 kg-meter^2

diag. I_2 = 0.1000, 0.7300, 0.8025 kg-meter^2

diag. I_3 = 0.0222, 0.2160, 0.2245 kg-meter^2

$\bar{s}_i = (\bar{x}_i, \bar{y}_i, \bar{z}_i)$ is a position vector of the centre of mass of link i with respect to the i^{th} coordinate system.

$\bar{s}_1$ = 0.0, 0.0, 0.073 ^{T} meter

$\bar{s}_2$ = -0.432, 0.0, 0.0 ^{T} meter

$\bar{s}_3$ = 0.0, 0.0, 0.1 ^{T} meter

m_i is the mass of i^{th} joint

m_1 = 2.27 kg, m_2 = 15.97 kg, m_3 = 11.36 kg.

APPENDIX II

LINK PARAMETERS AND MASS PROPERTIES OF PUMA 600 ROBOT

The link parameters associated with the six joints of the PUMA 600 robot arm are given below.

Joint number (i)	α_i in degrees	α_i in degrees	a_i in meters	d_i in meters
1	-160 to +160	-90	0	0
2	-225 to +45	0	0.86	0.25
3	-45 to +225	90	-0.04	0
4	-110 to +170	-90	0	0.86
5	-100 to +100	90	0	0
6	-266 to +266	0	0	0.3

The mass properties are specified as follows:

I_i is the inertia matrix for i^{th} joint. Then,

$$I_i = \begin{bmatrix} I_{ixx} & 0 & 0 \\ 0 & I_{iyy} & 0 \\ 0 & 0 & I_{izz} \end{bmatrix}$$

diag. I_1 = 4.5 4.5 0.56 kg-meter^2

diag. I_2 = 10.8 35.1 26.9 kg-meter^2

diag. I_3 = 14.1 14.1 0.74 kg-meter^2

diag. I_4 = 0.09 0.06 0.09 kg-meter^2

diag. I_5 = 0.006 0.009 0.006 kg-meter^2

diag. I_6 = 0.29 0.29 0.0061 kg-meter^2

$\bar{s}_i = (\bar{x}_i, \bar{y}_i, \bar{z}_i)$ is a position vector of the centre of mass of link i with respect to the i^{th} coordinate system.

$\bar{s}_1$ = 0 0 0.15 meter

$\bar{s}_2$ = -0.43 0 0 meter

$\bar{s}_3$ = 0 0 0.43 meter

$\bar{s}_4$ = 0 -0.03 0 meter

$\bar{s}_5$ = 0.006 0 0 meter

$\bar{s}_6$ = 0 0 -0.15 meter

m_i is the mass of i^{th} joint

m_1 = 49.9 kg m_2 = 109.0 kg m_3 = 57.0 kg

m_4 = 13.3 kg m_5 = 6.5 kg m_6 = 9.5 kg.

EE-1988-M-WAK-ON